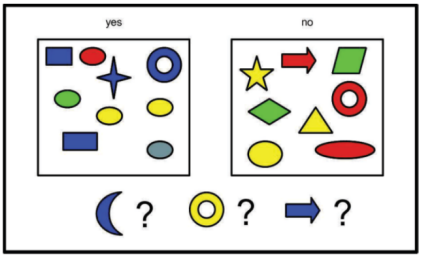
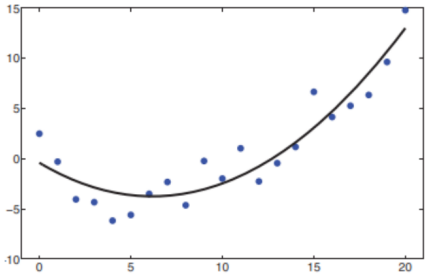
Classification (y_i binary or categorical)



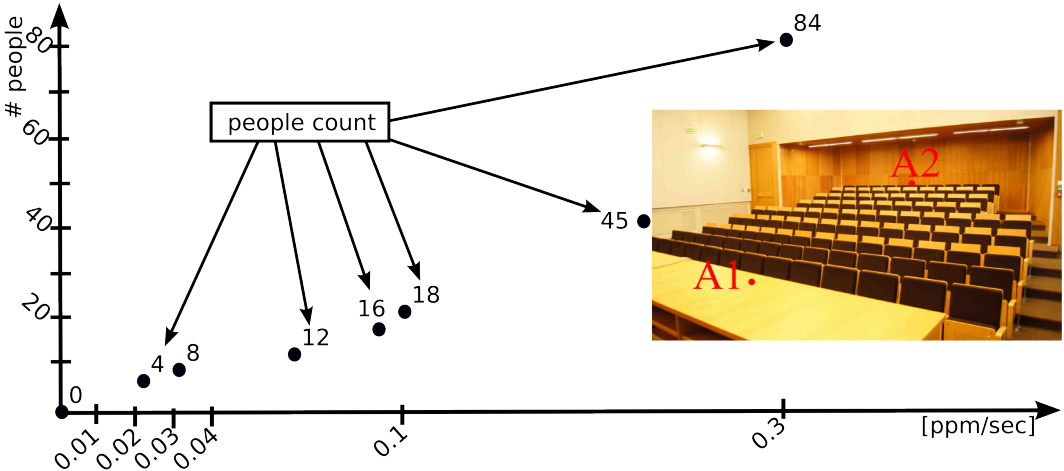
Murphy, Fig 1.1

Regression (y_i real-valued)

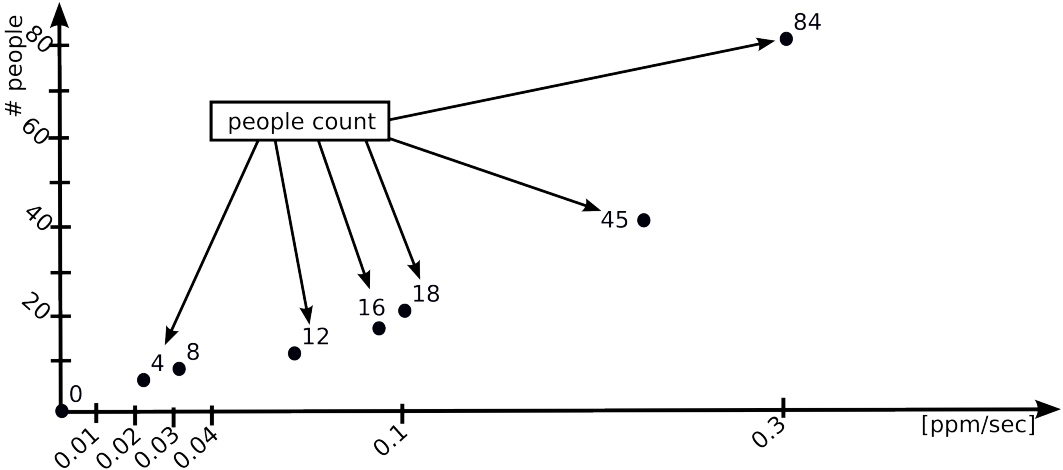


Murphy, Fig. 1.7

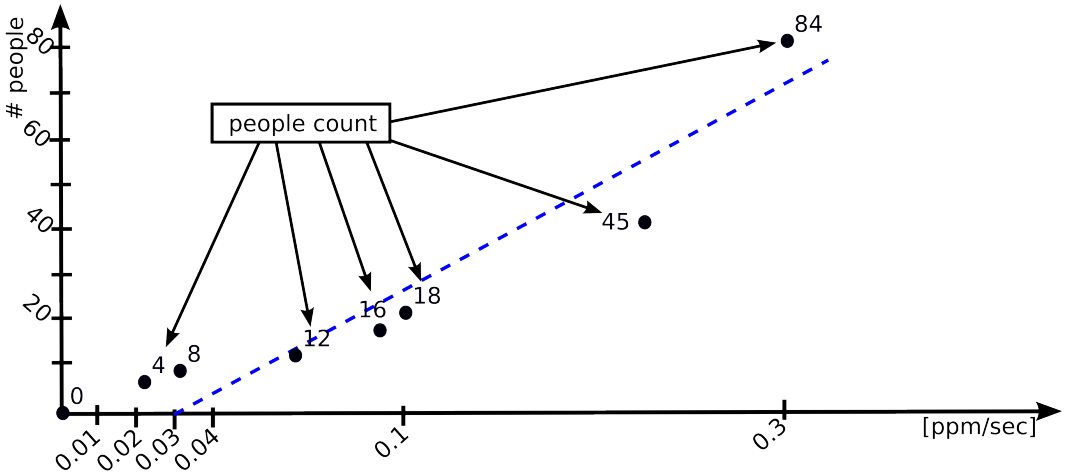
Linear regression



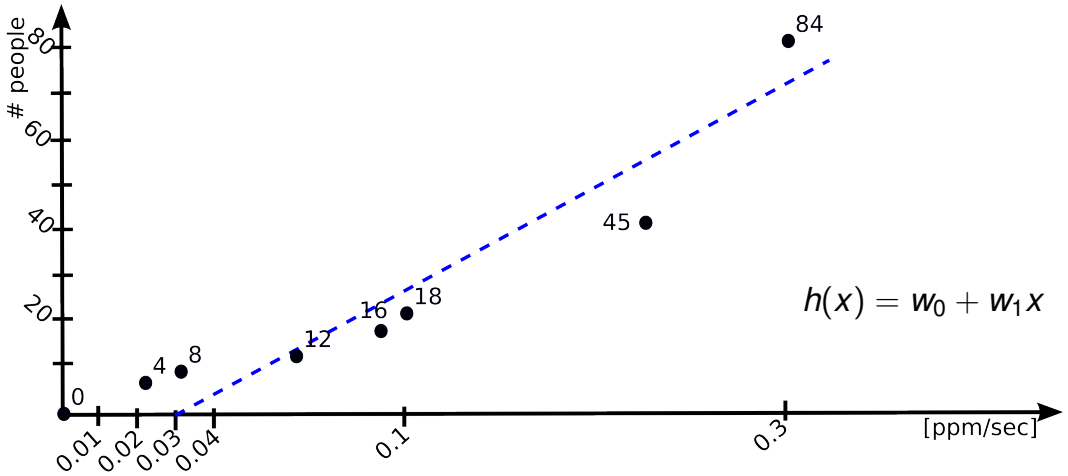
Linear regression



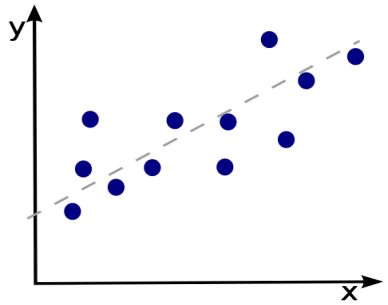
Linear regression



Linear regression



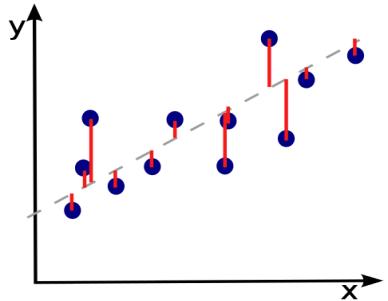
Linear regression



Objective function: $h(x) = w_0 + w_1 x$

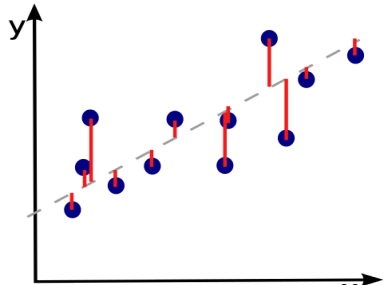
How to choose the parameter w_0 and w_1?

Linear regression



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

Linear regression – Loss function



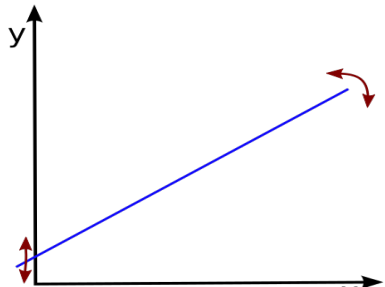
Some authors refer to the Loss function as *error function* or *cost function*. All these are synonyms.

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

$$\text{minimize } E[w_0, w_1] = L[(X, Y), h(x)] = \frac{1}{2n} \sum_{i=1}^{n} (h(x_i) - y_i)^2$$

Loss function estimates quality of current solution (Gradient descent).

Loss function – intuition

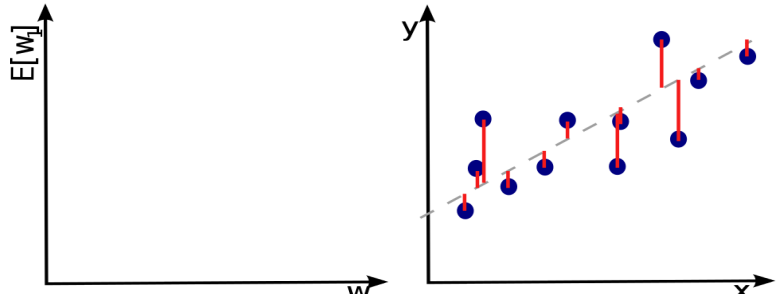


additive constant w_0 ignored in the following for simplicity

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

$$\text{minimize } E[w_0, w_1] = \frac{1}{2n} \sum_{i=1}^{n} (h(x_i) - y_i)^2$$

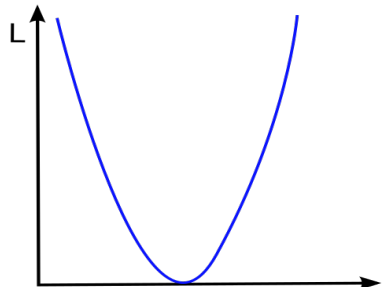
Loss function – intuition



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

$$\text{minimize } E[w_0, w_1] = \frac{1}{2n} \sum_{i=1}^{n} (h(x_i) - y_i)^2$$

Loss function – intuition

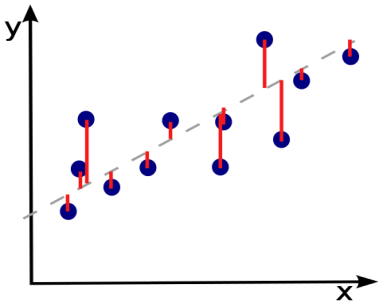


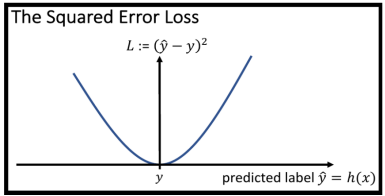
additive constant w_0 ignored in this figure

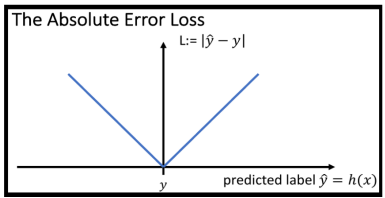
$$h(x) = w_0 + w_1 x$$

$$\text{minimize } E[w_0, w_1] = \frac{1}{2n} \sum_{i=1}^{n} (h(x_i) - y_i)^2$$

Common loss functions

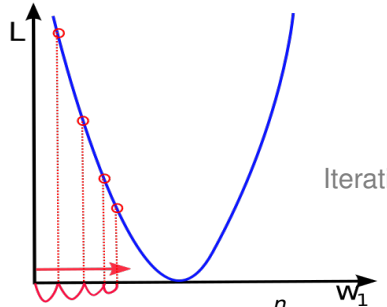






Hypothesis: $h(x) = w_0 + w_1 x$

Gradient Descent

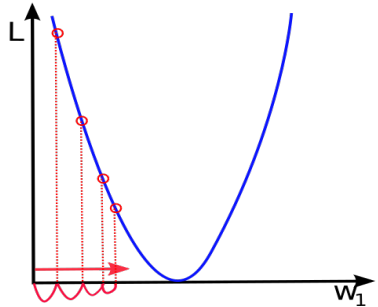


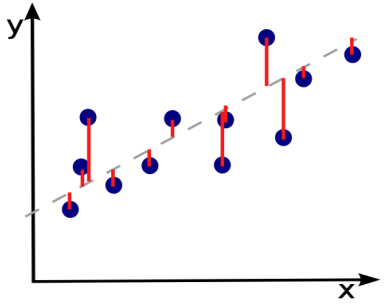
Iterative approximation of w_1

$$\text{minimize } E[w_0, w_1] = \frac{1}{2n}\sum_{i=1}^{n}(h(x_i) - y_i)^2$$

$$w_1 = w_1 - \delta \cdot \frac{\partial}{\partial w_1} E[w_0, w_1]$$

Gradient descent loss function





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

$$\text{minimize } E[w_0, w_1] = \frac{1}{2n} \sum_{i=1}^{n} (h(x_i) - y_i)^2$$

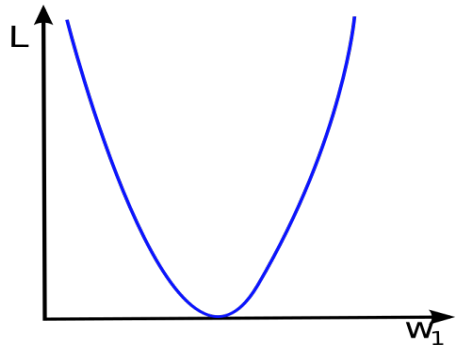
Least squares estimation

Given a loss function

$$E[w_0, w_1] = \frac{1}{2n} \sum_{i=1}^{n} (y_i - (w_0 + w_1 x_i))^2$$

we can minimize the error by requiring

$$\frac{\partial E}{\partial w_0} = 0, \frac{\partial E}{\partial w_1} = 0$$



Least squares estimation

Given a loss function

$$E[w_0, w_1] = \frac{1}{2n} \sum_{i=1}^{n} (y_i - (w_0 + w_1 x_i))^2$$

we can minimize the error by requiring

$$\frac{\partial E}{\partial w_0} = 0, \frac{\partial E}{\partial w_1} = 0$$

Differentiation yields

$$\begin{aligned} \frac{\partial E}{\partial w_0} &= \textstyle\sum_{i=1}^{n} \frac{2}{2n} (y_i - (w_1 x_i + w_0)) \cdot 1 \\ \frac{\partial E}{\partial w_1} &= \textstyle\sum_{i=1}^{n} \frac{2}{2n} (y_i - (w_1 x_i + w_0)) \cdot x_i \end{aligned}$$

$$\frac{\partial E}{\partial w_0} = \sum_{i=1}^{n} \frac{2}{2n} \left(y_i - (w_1 x_i + w_0)\right) \cdot 1$$
$$\frac{\partial E}{\partial w_1} = \sum_{i=1}^{n} \frac{2}{2n} \left(y_i - (w_1 x_i + w_0)\right) \cdot x_i$$

Setting

$$\frac{\partial E}{\partial w_0} = \frac{\partial E}{\partial w_1} = 0$$

will lead to

$$\sum_{i=1}^{n} \left(y_i - (w_1 x_i + w_0)\right) = 0$$
$$\sum_{i=1}^{n} \left(y_i - (w_1 x_i + w_0)\right) \cdot x_i = 0$$

Least squares estimation

$$\sum_{i=1}^{n} (y_i - (w_1 x_i + w_0)) = 0$$

$$\sum_{i=1}^{n} (y_i - (w_1 x_i + w_0)) \cdot x_i = 0$$

rewrite as

$$\left(\sum_{i=1}^{n} x_i\right) w_1 + \left(\sum_{i=1}^{n} 1\right) w_0 = \sum_{i=1}^{n} y_i$$

$$\left(\sum_{i=1}^{n} x_i^2\right) w_1 + \left(\sum_{i=1}^{n} x_i\right) w_0 = \sum_{i=1}^{n} x_i y_i$$

Least squares estimation

$$\left(\sum_{i=1}^{n} x_i\right) w_1 + \left(\sum_{i=1}^{n} 1\right) w_0 = \sum_{i=1}^{n} y_i$$
$$\left(\sum_{i=1}^{n} x_i^2\right) w_1 + \left(\sum_{i=1}^{n} x_i\right) w_0 = \sum_{i=1}^{n} x_i y_i$$

Consequently, values of w_0 and w_1 that minimize the error satisfy

$$\begin{pmatrix} \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \end{pmatrix} \begin{pmatrix} w_1 \\ w_0 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{pmatrix}$$

Least squares estimation

$$\begin{pmatrix} \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \end{pmatrix} \begin{pmatrix} w_1 \\ w_0 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{pmatrix}$$

for an invertible matrix this implies

$$\begin{pmatrix} w_1 \\ w_0 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \end{pmatrix}^{-1} \begin{pmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{pmatrix}$$

Solve this linear equation system to find optimal values for w_0 and w_1.

Least squares estimation

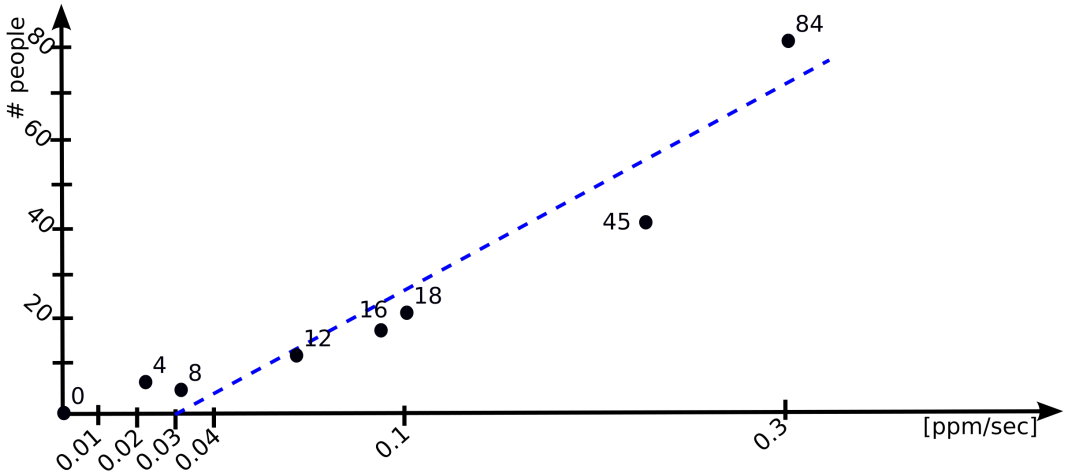
$$\begin{pmatrix} \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \end{pmatrix} \begin{pmatrix} w_1 \\ w_0 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{pmatrix}$$

for an invertible matrix this implies

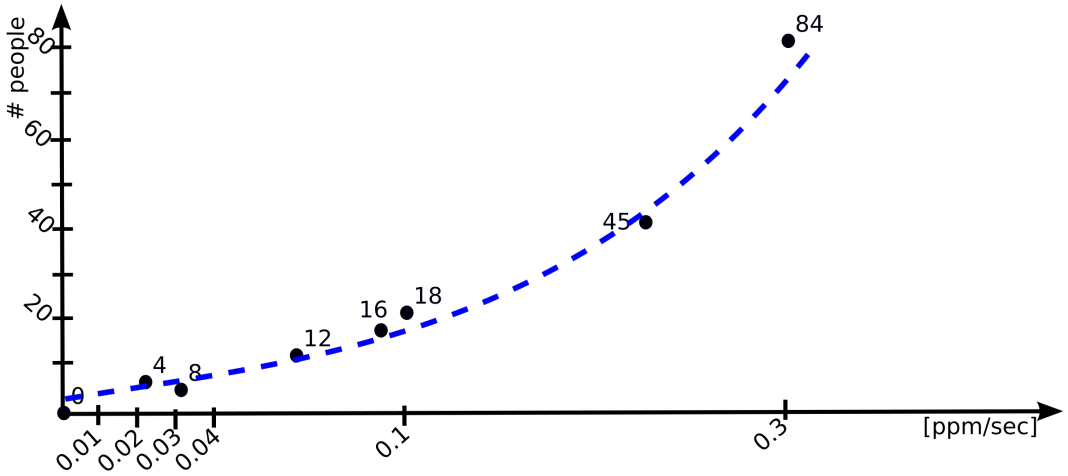
$$\begin{pmatrix} w_1 \\ w_0 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \end{pmatrix}^{-1} \begin{pmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{pmatrix}$$

Solve this linear equation system to find optimal values for w_0 and w_1.
Note: matrix must be invertible.

Polynomial regression



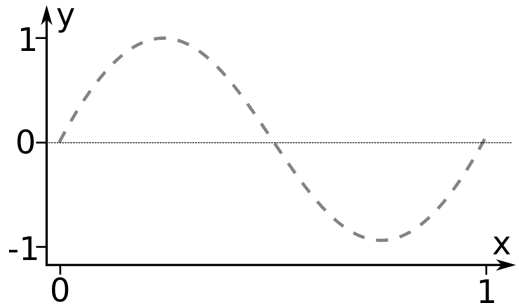
Polynomial regression



Polynomial regression (Polynomial curve fitting)

Example

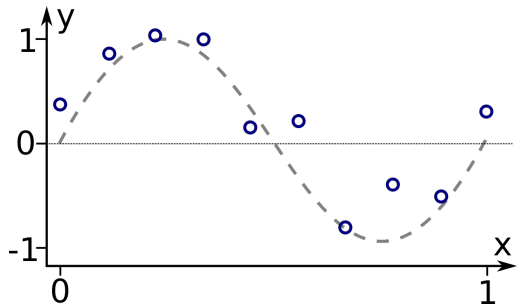
Sample points are created for the function $\sin(2\pi x) + \mathcal{N}$ where $\mathcal{N}$ is a random noise value



Polynomial regression (Polynomial curve fitting)

Example

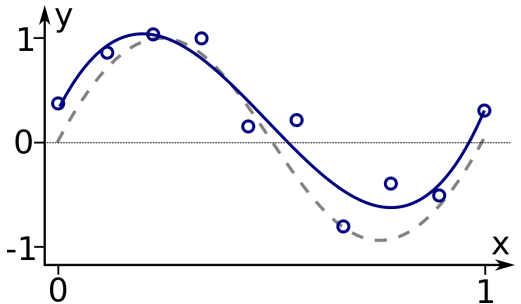
Sample points are created for the function $\sin(2\pi x) + \mathcal{N}$ where $\mathcal{N}$ is a random noise value



Polynomial regression (Polynomial curve fitting)

Example

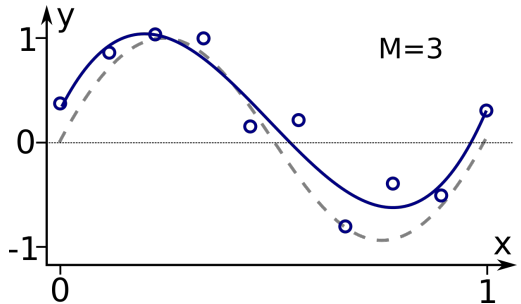
Sample points are created for the function $\sin(2\pi x) + \mathcal{N}$ where $\mathcal{N}$ is a random noise value



Polynomial curve fitting

We fit the data points into a polynomial function:

$$h(x, \vec{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$



Polynomial curve fitting

We fit the data points into a polynomial function:

$$h(x, \vec{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

This can be obtained by minimising a loss function which measures the misfit between $h(x, \vec{w})$ and the training data set:

$$E(\vec{w}) = \frac{1}{2n} \sum_{i=1}^{n} \left[h(x_i, \vec{w}) - y_i \right]^2$$

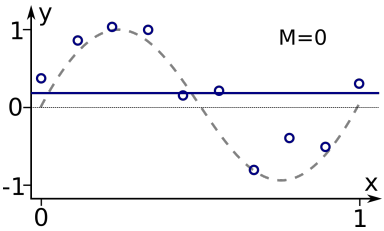
$E(\vec{w}) \geq 0$; $E(\vec{w}) = 0$ IFF all points are covered by the function

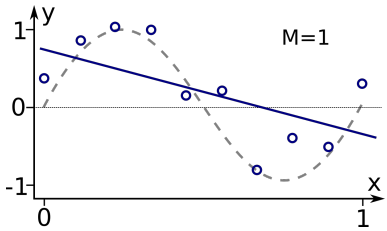
Polynomial curve fitting

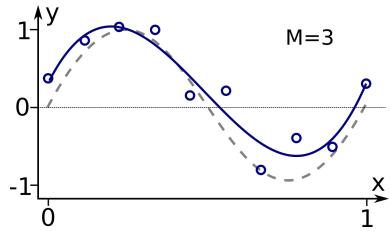
One problem is the right choice of the dimension *M*

When M is too small, the approximation accuracy might be bad

$$h(x, \vec{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$



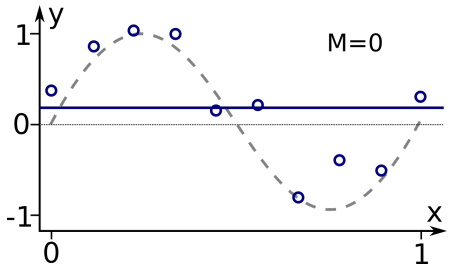


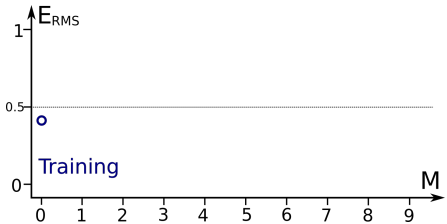


Polynomial curve fitting

Visualise error $E(\vec{w})$ wrt the data by Root of the Mean Squared (RMS)

$$E_{RMS} = \sqrt{\frac{2E(\vec{w})}{n}}$$

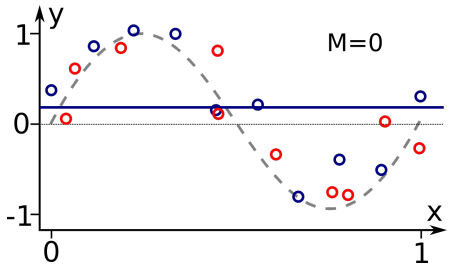


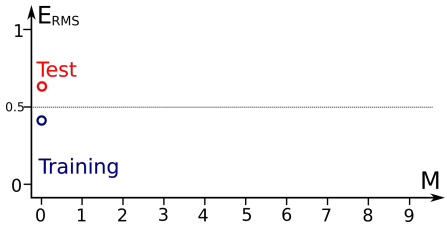


Polynomial curve fitting

Visualise error $E(\vec{w})$ wrt the data by Root of the Mean Squared (RMS)

$$E_{RMS} = \sqrt{\frac{2E(\vec{w})}{n}}$$

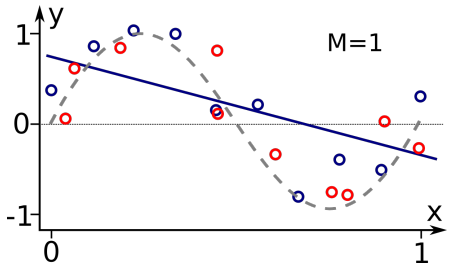


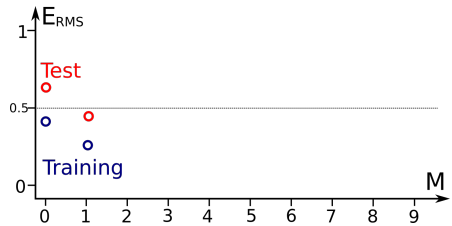


Polynomial curve fitting

Visualise error $E(\vec{w})$ wrt the data by Root of the Mean Squared (RMS)

$$E_{RMS} = \sqrt{\frac{2E(\vec{w})}{n}}$$

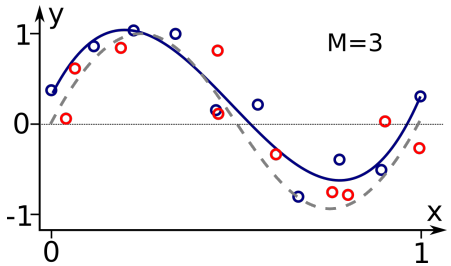


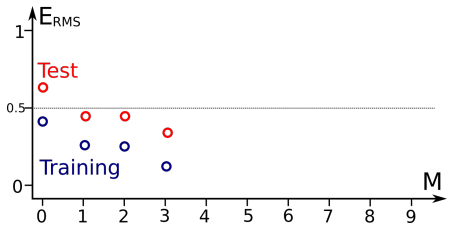


Polynomial curve fitting

Visualise error $E(\vec{w})$ wrt the data by Root of the Mean Squared (RMS)

$$E_{RMS} = \sqrt{\frac{2E(\vec{w})}{n}}$$

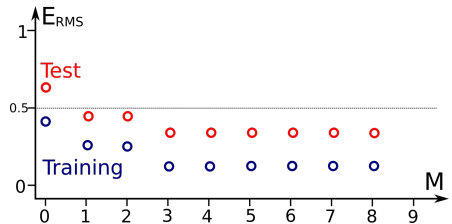




Polynomial curve fitting

Visualise error $E(\vec{w})$ wrt the data by Root of the Mean Squared (RMS)

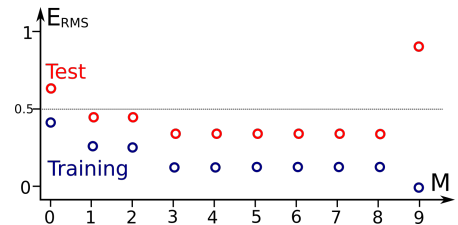
$$E_{RMS} = \sqrt{\frac{2E(\vec{w})}{n}}$$



Polynomial curve fitting

Visualise error $E(\vec{w})$ wrt the data by Root of the Mean Squared (RMS)

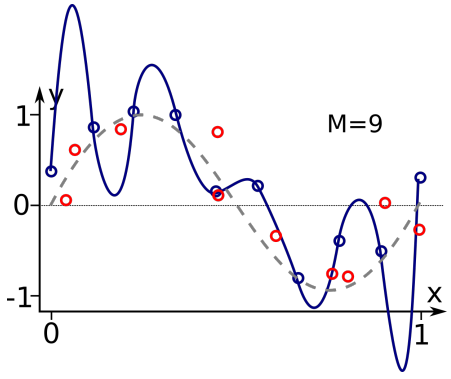
$$E_{RMS} = \sqrt{\frac{2E(\vec{w})}{n}}$$

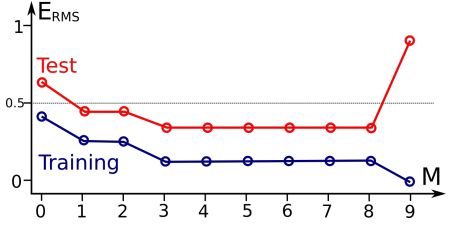


Polynomial curve fitting

Visualise error $E(\vec{w})$ wrt the data by Root of the Mean Squared (RMS)

$$E_{RMS} = \sqrt{\frac{2E(\vec{w})}{n}}$$



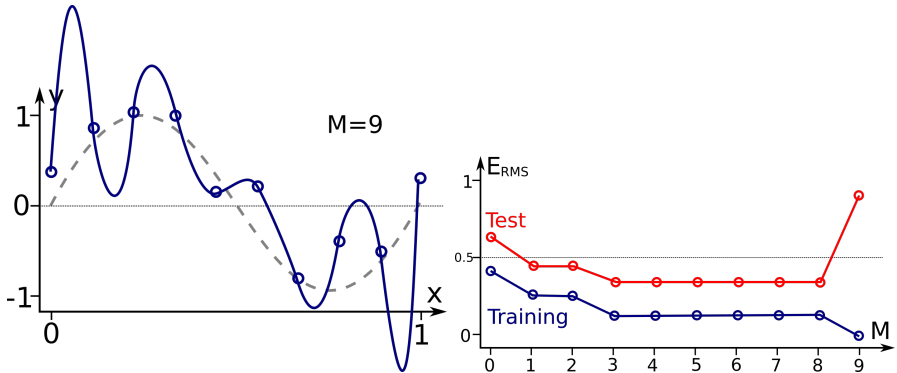


Polynomial curve fitting

This event is called overfitting

The polynomial is now trained too well to the training data

It performs badly on test data



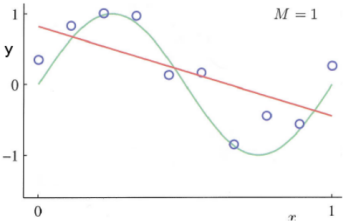
Model selection

High Bias
(underfitting)

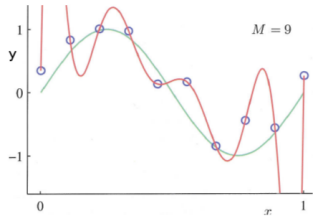
High Variance
(overfitting)

Model selection

High Bias
(underfitting)

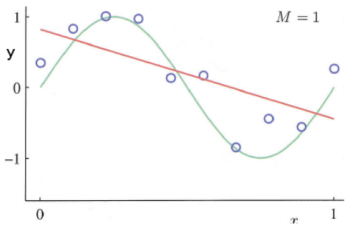


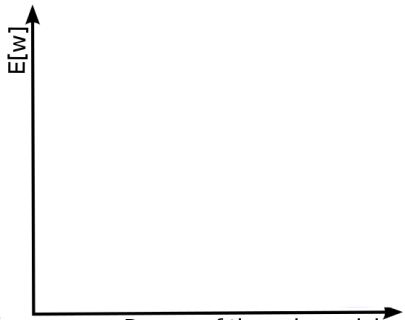
High Variance
(overfitting)



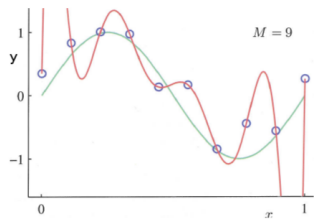
Model selection

High Bias
(underfitting)

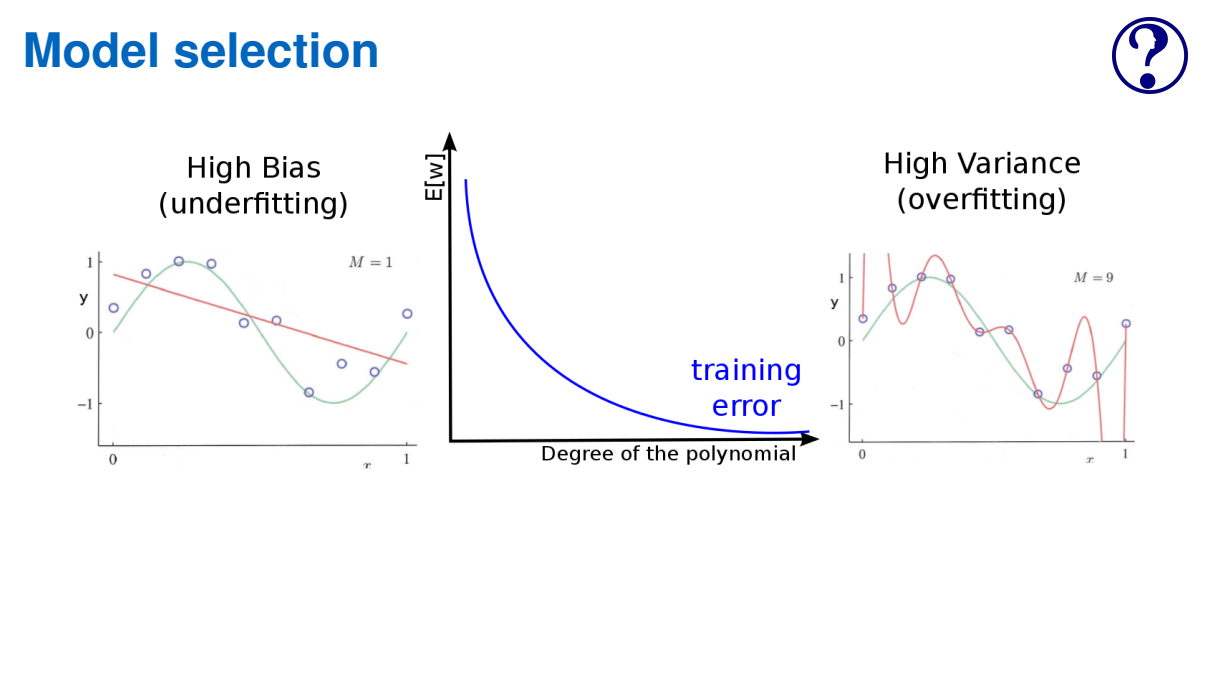




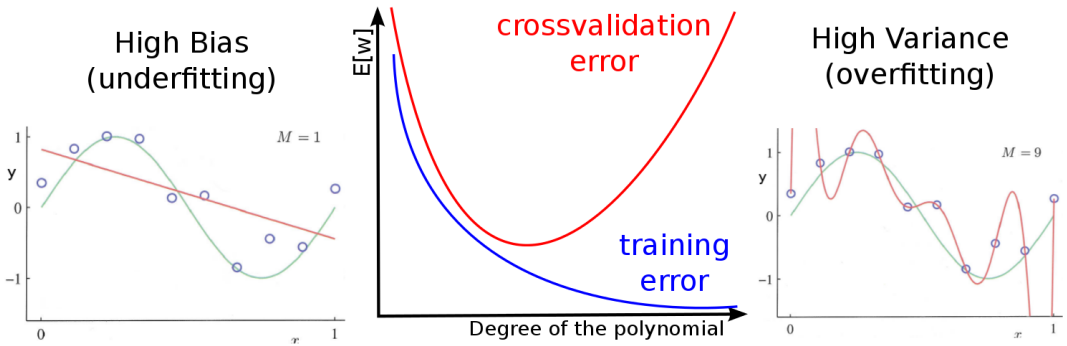
High Variance
(overfitting)



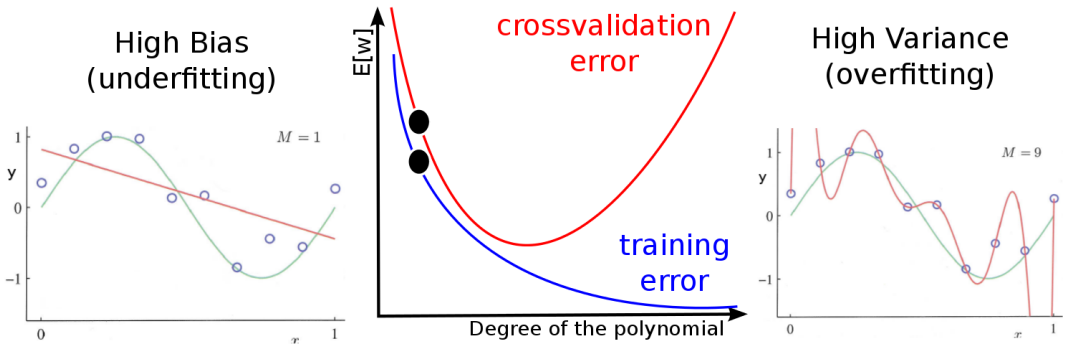
Model selection



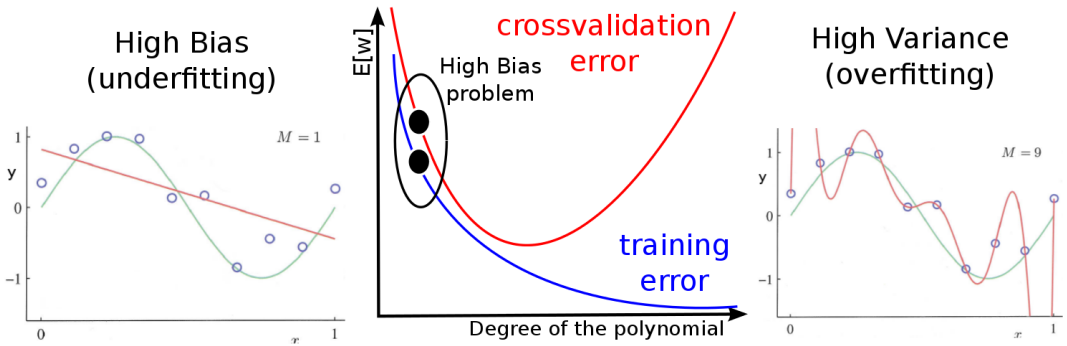
Model selection



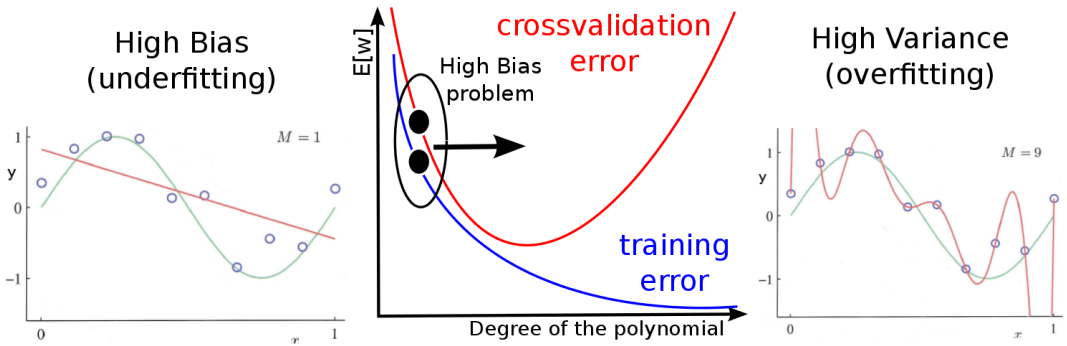
Model selection



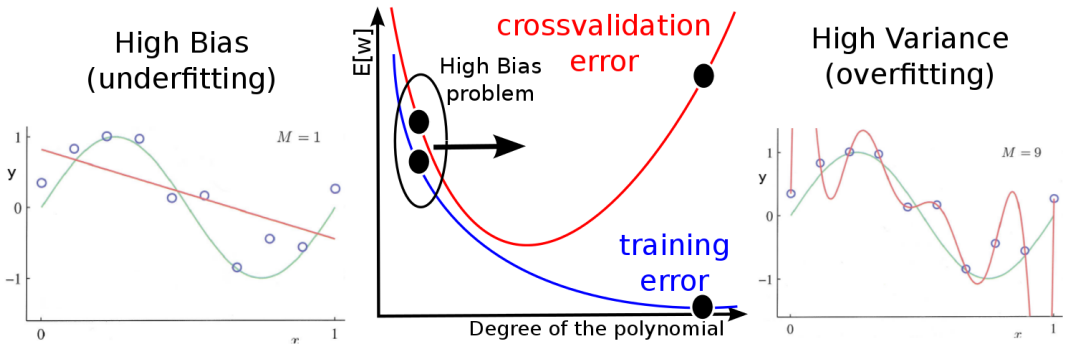
Model selection



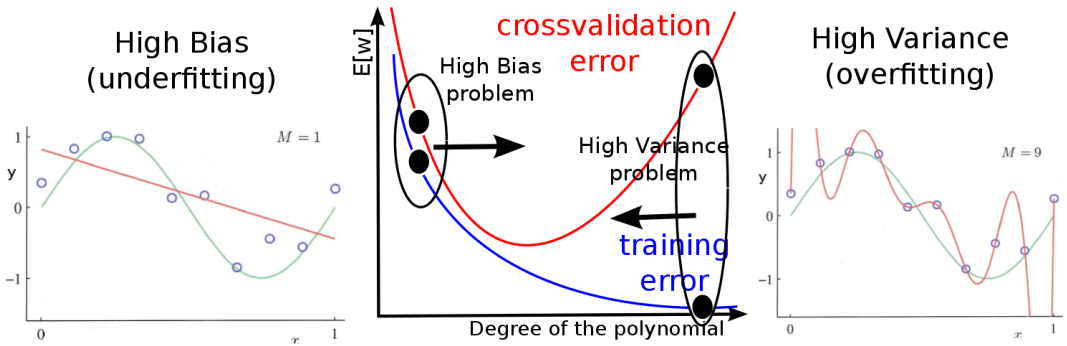
Model selection



Model selection



Model selection



Polynomial curve fitting

One solution to cope with overfitting is regularisation

A penalty term is added to the loss function

This term discourages the coefficients $\vec{w}$ from reaching large values

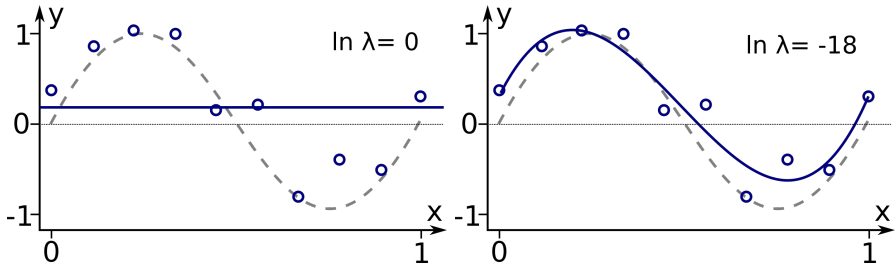
$$\overline{E}(\vec{w}) = \frac{1}{2n}\sum_{i=1}^{n}\left[h(x_i, \vec{w}) - y_i\right]^2 + \frac{\lambda}{2}||\vec{w}||^2$$

with

$$||\vec{w}||^2 = \vec{w}^T\vec{w} = w_0^2 + w_1^2 + \cdots + w_M^2$$

Polynomial curve fitting

Depending on the value of λ, overfitting is controlled



$$\overline{E}(\vec{w}) = \frac{1}{2n}\sum_{i=1}^{n}\left[h(x_i, \vec{w}) - y_i\right]^2 + \frac{\lambda}{2}||\vec{w}||^2$$

Learning curves

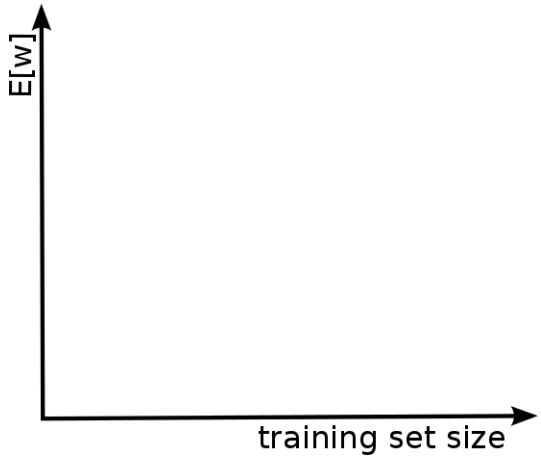
Learning Curves

Plotting learning curves helps to find out, whether our algorithm suffers from high variance or high bias

Learning curves

Learning Curves

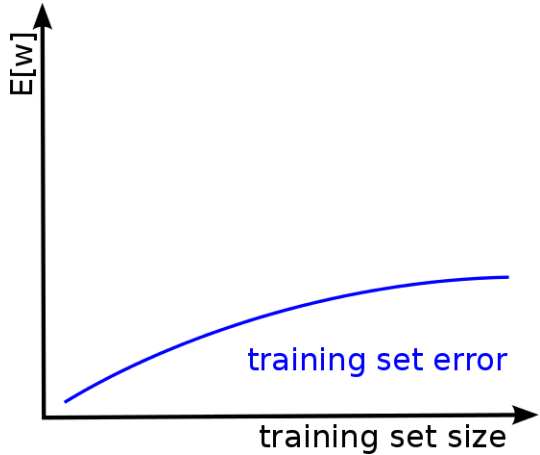
Plotting learning curves helps to find out, whether our algorithm suffers from high variance or high bias



Learning curves

Learning Curves

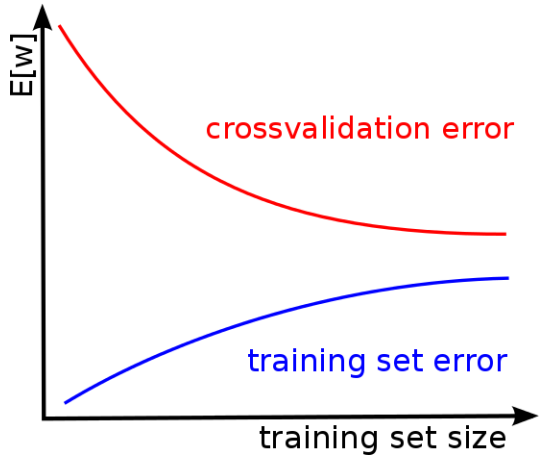
Plotting learning curves helps to find out, whether our algorithm suffers from high variance or high bias



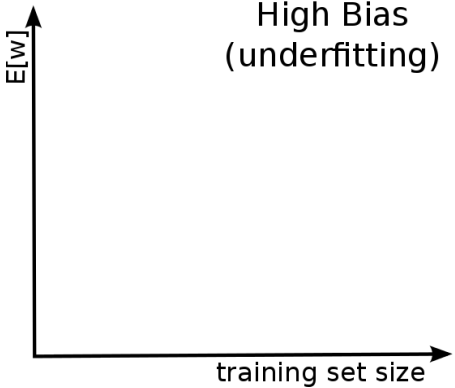
Learning curves

Learning Curves

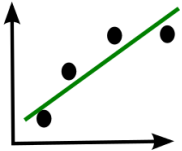
Plotting learning curves helps to find out, whether our algorithm suffers from high variance or high bias

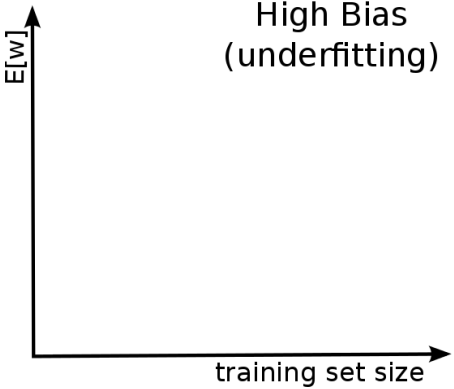


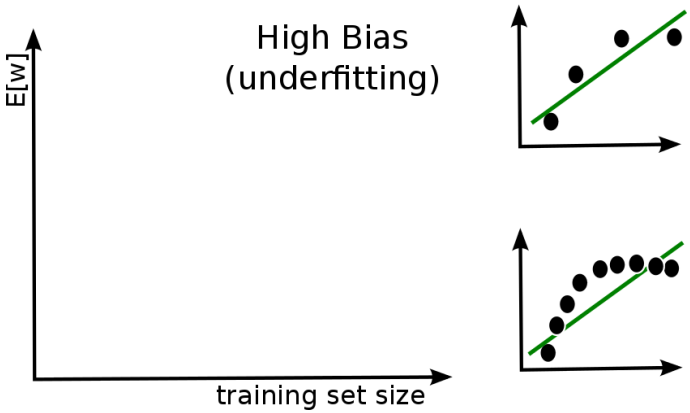
High Bias
(underfitting)

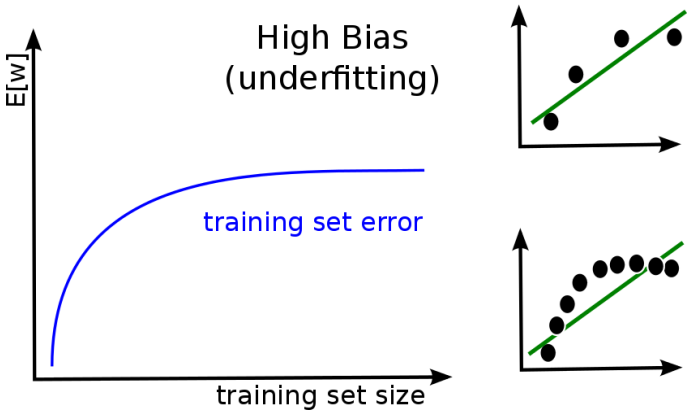


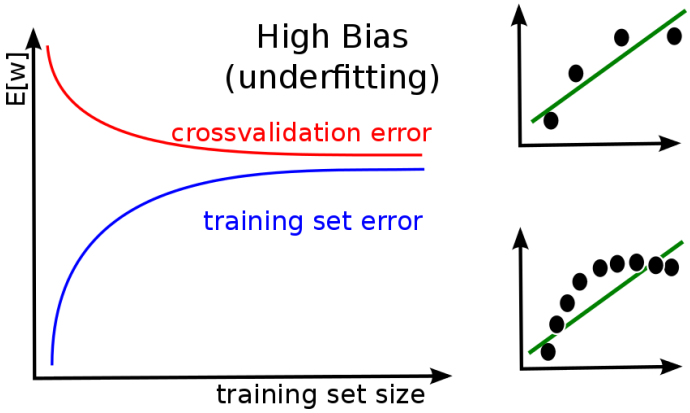
High Bias
(underfitting)

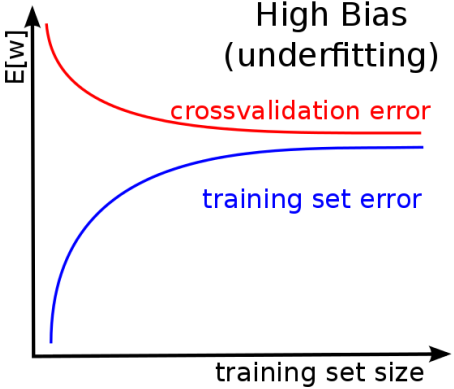


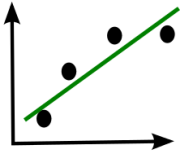


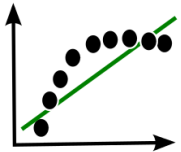

High Bias
(underfitting)
E[w]
training set size


High Bias
(underfitting)
E[w]
training set error
training set size


High Bias
(underfitting)
E[w]
crossvalidation error
training set error
training set size



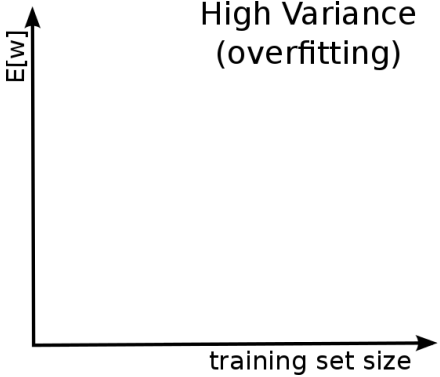




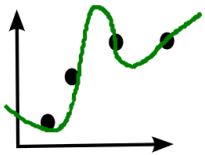
When the algorithm suffers from high Bias...

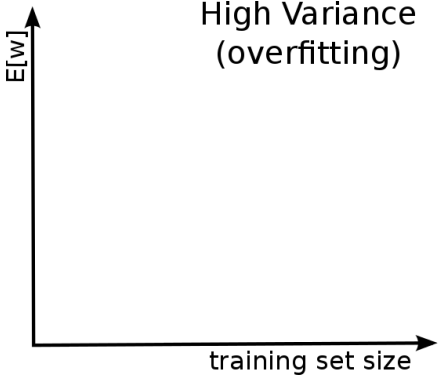
- → crossvalidation error and training error are close
- → Increasing the training set size does not help !

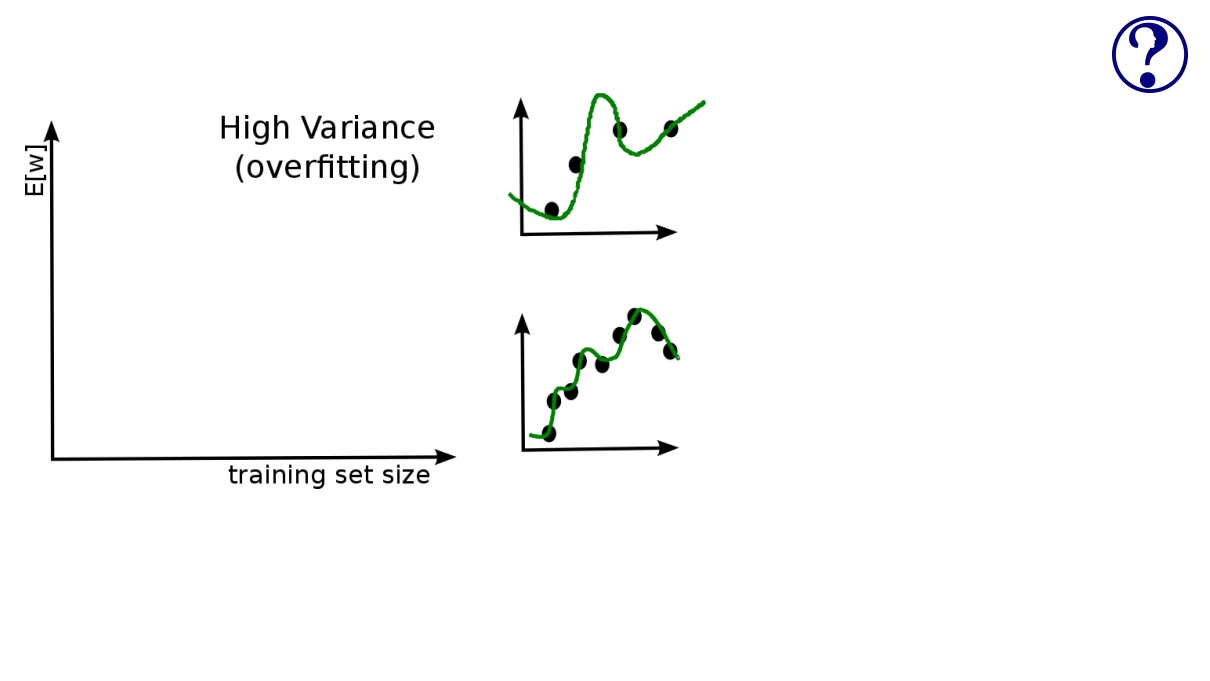
High Variance
(overfitting)

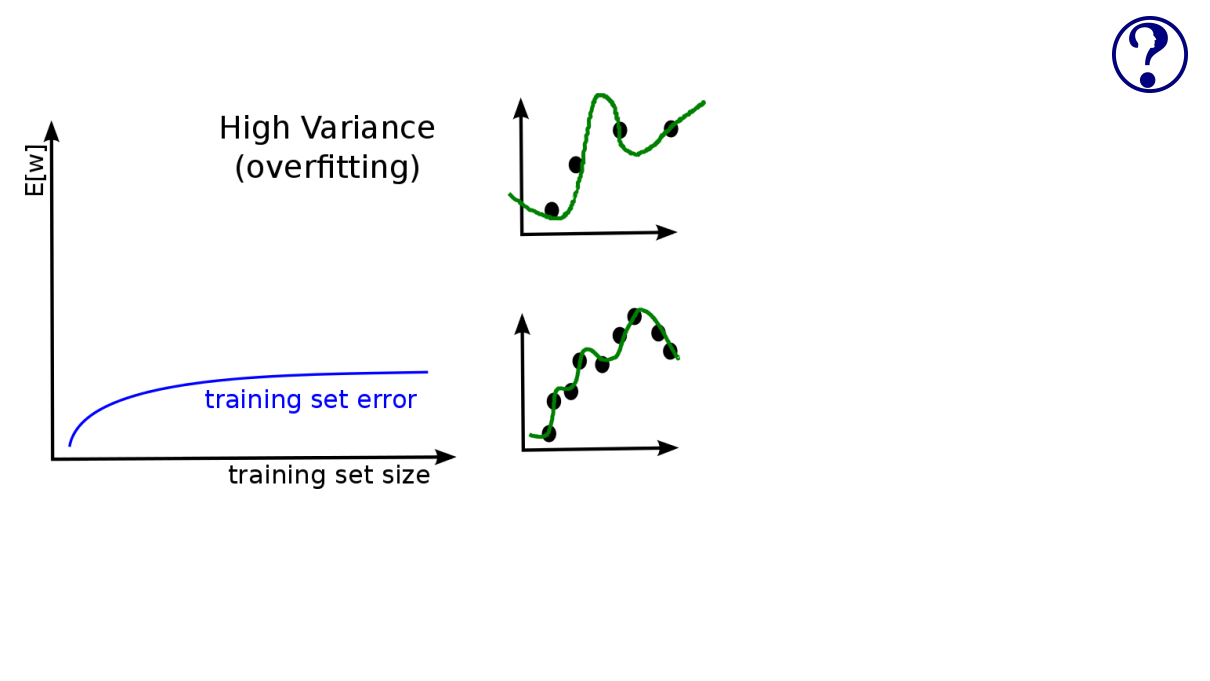


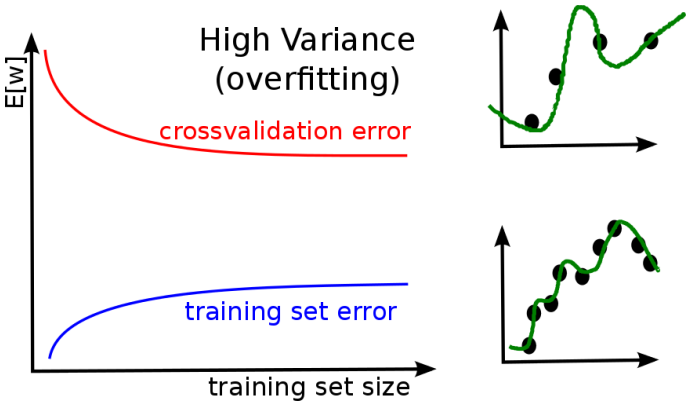
High Variance
(overfitting)

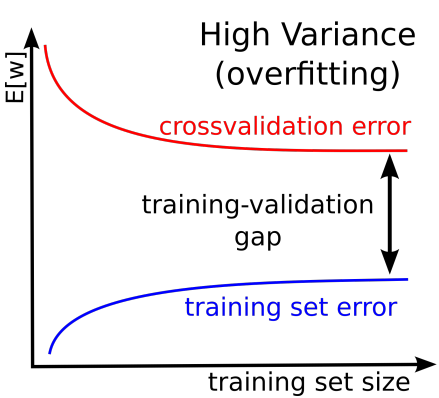


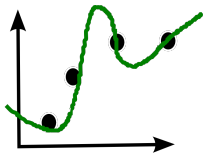


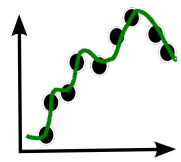

High Variance
(overfitting)
E[w]
training set size


High Variance
(overfitting)
E[w]
training set error
training set size


High Variance
(overfitting)
E[w]
crossvalidation error
training set error
training set size







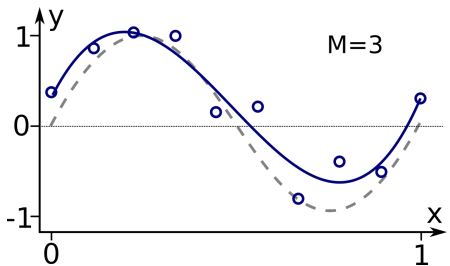
When the algorithm suffers from high variance...

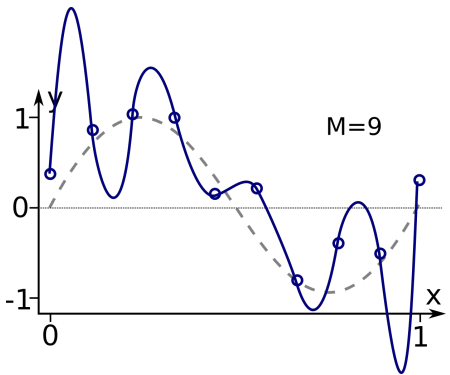
- → crossvalidation error and training error are far apart
- → Increasing the training set size improves the performance

Polynomial curve fitting

When M becomes too big, the polynomial will cross all points exactly

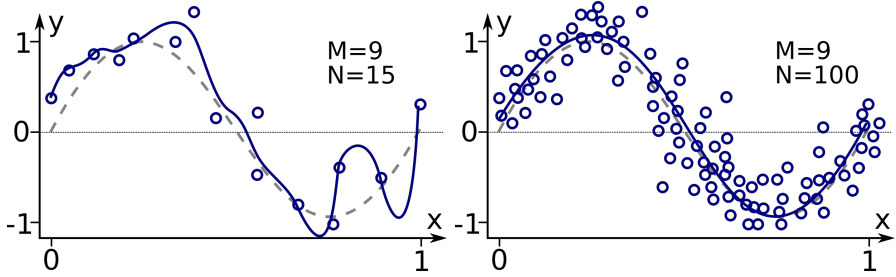
For $M = n$, it is always possible to create a polynomial of order M that contains all values in the data set.





Polynomial curve fitting

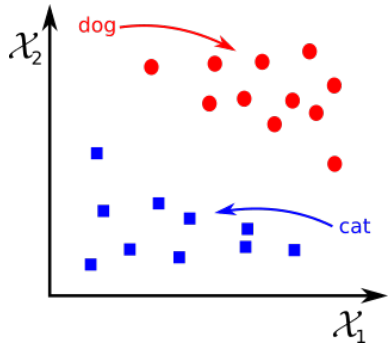
With increasing number of data points, the problem of overfitting becomes less severe for a given value of *M*

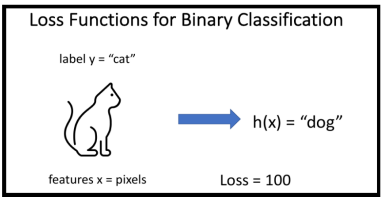


Logistic regression

Nominal classes

Classes might be nominal in real-world problems

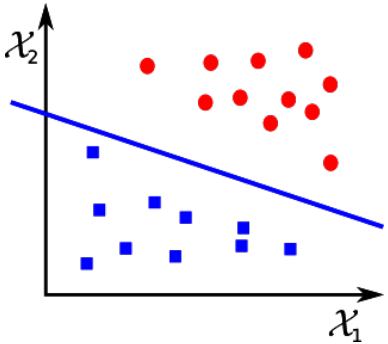


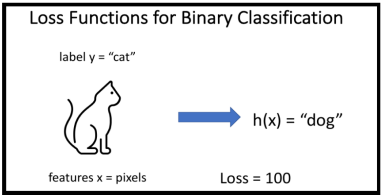


Logistic regression

Nominal classes

Classes might be nominal in real-world problems



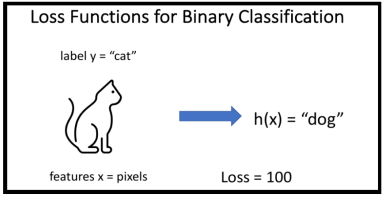


Logistic regression

Nominal classes

Classes might be nominal in real-world problems

Weather	Sunny, rainy
Medical	positive diagnosis, negative diagnosis
Localisation	indoor, outdoor



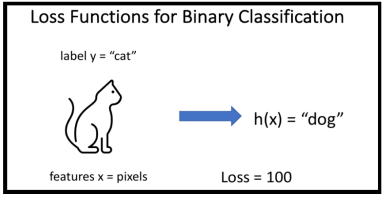
Logistic regression

Nominal classes

Classes might be nominal in real-world problems

- Weather: Sunny, rainy
- Medical: positive diagnosis, negative diagnosis
- Localisation: indoor, outdoor

In such case, classification is binary: $y \in \{0, 1\}$



Logistic regression

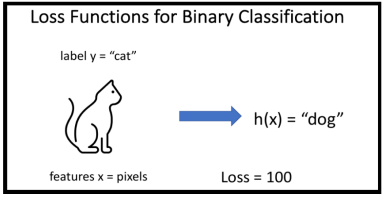
Nominal classes

Classes might be nominal in real-world problems

- Weather: Sunny, rainy
- Medical: positive diagnosis, negative diagnosis
- Localisation: indoor, outdoor

In such case, classification is binary: $y \in \{0, 1\}$

Linear regression: $h(x)$ can be smaller than 0 or greater than 1



Logistic regression

Nominal classes

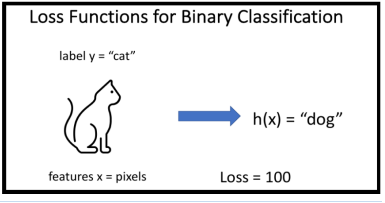
Classes might be nominal in real-world problems

- Weather: Sunny, rainy
- Medical: positive diagnosis, negative diagnosis
- Localisation: indoor, outdoor

In such case, classification is binary: $y \in \{0, 1\}$

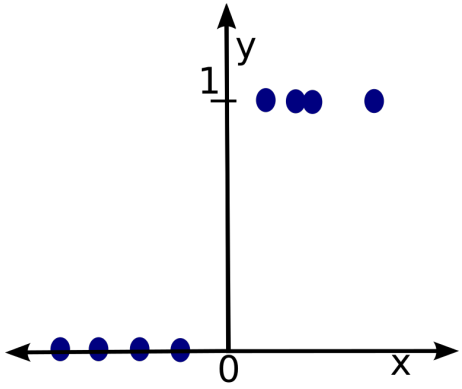
Linear regression: $h(x)$ can be smaller than 0 or greater than 1

Logistic regression: $0 \leq h(x) \leq 1$



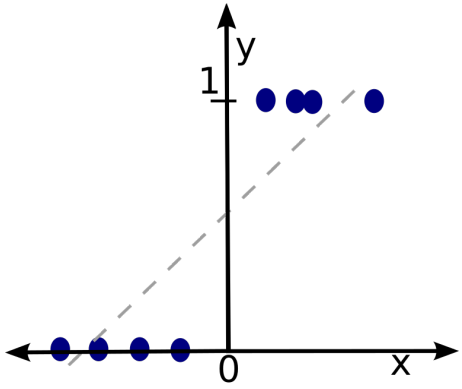
Logistic regression

Loss function



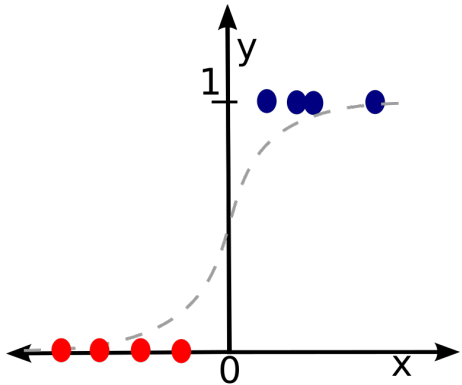
Logistic regression

Loss function



Logistic regression

Loss function



Logistic regression

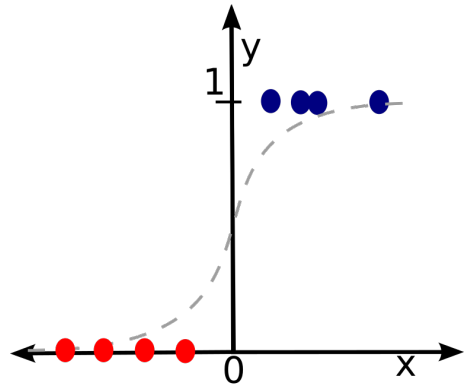
Loss function

Linear regression

$$h(x) = \vec{w}^T x$$

Logistic regression

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



Logistic regression

Loss function

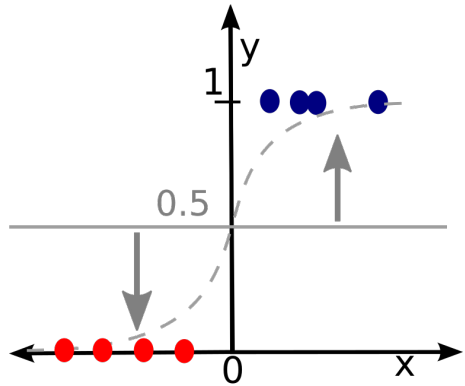
Linear regression

$$h(x) = \vec{w}^T x$$

Logistic regression

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

$$y = \begin{cases} 1 & \text{if } h(x) \geq 0.5 \\ 0 & \text{else} \end{cases}$$

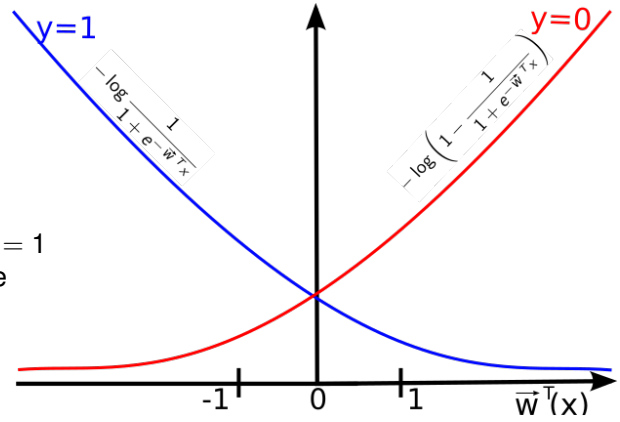


Logistic regression

Loss function

$$y = \begin{cases} 1 & \text{if } h(x) \geq 0.5 \\ 0 & \text{else} \end{cases}$$

$$L[(\mathcal{X}, \mathcal{Y}), h(\cdot)] = \begin{cases} -\log(h(x)) & \text{if } y = 1 \\ -\log(1 - h(x)) & \text{else} \end{cases}$$



Questions?

Literature

- C.M. Bishop: Pattern recognition and machine learning, Springer, 2007.
- R.O. Duda, P.E. Hart, D.G. Stork: Pattern Classification, Wiley, 2001.