UNIVERSITY OF LEEDS

Class: Machine Learning

Linear Models for Regression

Instructor: Matteo Leonetti

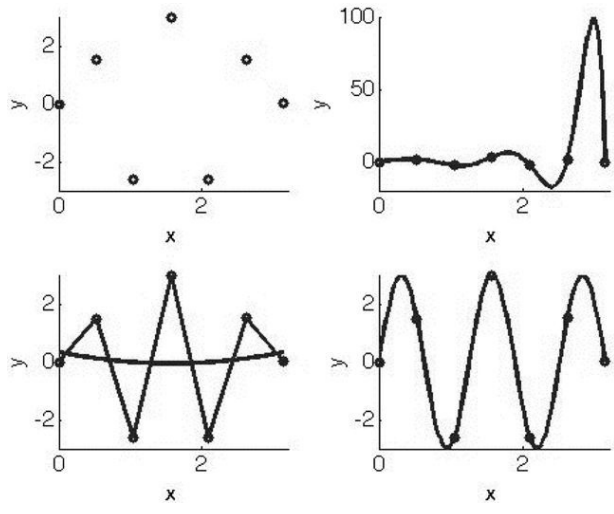
Learning outcomes

UNIVERSITY OF LEEDS

- Approximate a function through linear regression with different basis functions
- Compute the pseudo-inverse of a matrix
- Derive the bias-variance decomposition

Back to regression

UNIVERSITY OF LEEDS



We now look at a different supervised learning problem: regression.

Regression is the problem of approximating a function from a set of sample points.

Linear regression

UNIVERSITY OF LEEDS

Unknown function $t = f(\boldsymbol{x})$

Samples: $\langle \boldsymbol{x}_i, t_i \rangle$

We approximate the function with a linear model (in the parameters):

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 x_1 + w_2 x_3 + \cdots + w_M x_M$$

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + \sum_{i=1}^{M} w_i \phi_i(x)$$

Basis functions:

In linear regression, the model used for the function is linear in the parameters (hence the name).

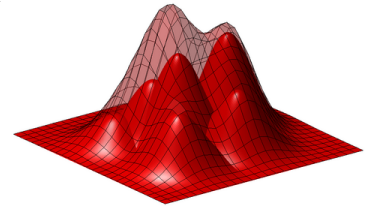
However, it does not need to be linear in the input, since non-linear functions can be obtained with the use of basis functions.

Basis functions

UNIVERSITY OF LEEDS

Polynomial basis: $\phi_i(x) = x^i$

$$y(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M$$



"Gaussian" basis: $\phi_i(x) = \exp\{-\frac{(x-\mu_i)^2}{2s^2}\}$

sigmoid basis: $\phi_i(x) = \sigma(\frac{x-\mu}{s}) = \sigma_{\mu,s}$

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

A few examples of possible basis functions

The polynomial bases, for instance, allows to represent polynomials of any order rather than only hyperplanes, as if the function was also linear in the input.

Another simple but very important example is the gaussian basis, where the learned function is a combination of different gaussians uniformly placed in the domain. Each gaussian provides a "bump" and by multiplying each bump by a weight we either increase of decrease its effect. Combining the different bumps can give us almost any shape.

System of equations

UNIVERSITY OF LEEDS

Samples: $\langle \boldsymbol{x}_i, t_i \rangle = \langle x_{i,1}, x_{i,2}, \cdots, x_{i,m}, t_i \rangle$

Let's set: $\phi_0 = 1$ So that: $y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + \sum_{i=1}^{M} w_i \phi_i(x) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})$

For each point: $\boldsymbol{x}_j$ We denote: $\phi_i(\boldsymbol{x}_j) = \Phi_{i,j}$

Each point imposes a constraint:

$$\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) = t_i$$

For N points in the data set, we have N equations.

Can we find a weight vector that satisfies all the constraints?

For each sample we have a point in input space x, and the corresponding value t of the function we are trying to approximate.

If our approximation was exact, $w^T\Phi = t$ for each point. However, this is only possible if we have exactly as many parameters as unknowns, and this is generally not the case.

An Example - data

UNIVERSITY OF LEEDS

$\langle -2, 18.28 \rangle$

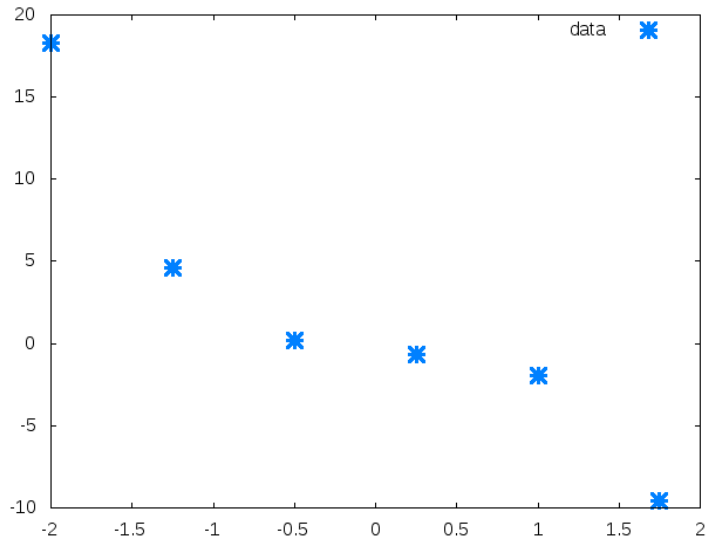
$\langle -1.25, 4.64 \rangle$

$\langle -0.5, 0.16 \rangle$

$\langle 0.25, -0.63 \rangle$

$\langle 1.75, -9.56 \rangle$

$\langle 1, -1.95 \rangle$



Let's look at an example. These points have been sampled from a certain function we want to approximate.

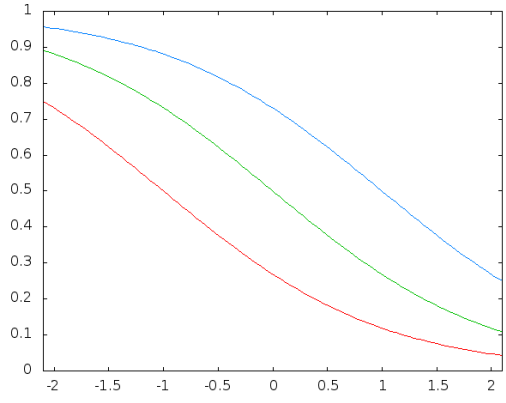
Choose model

UNIVERSITY OF LEEDS

How many sigmoids? Centred where? $y = ?$

Let's use these three:

$$y = w_0 + w_1 \sigma_{-1,1}(x) + w_2 \sigma_{0,1}(x) + w_3 \sigma_{1,1}(x)$$



The first step consists in choosing the basis functions.

This model choice is up to the designer (as you have practised and experimented in the coursework with neural networks).

Here I have chosen three sigmoids with different parameters completely arbitrarily, and I don't know how well I'll be able to approximate the target function with it. We'll see.

Equations

UNIVERSITY OF LEEDS

We create a system of equations by evaluating the basis functions on the data points:

$$w_0 + w_1 \sigma_{-1,1}(x) + w_2 \sigma_{0,1}(x) + w_3 \sigma_{1,1}(x) = t$$

$\Phi_{1,3}$

$\langle -2, 18.28 \rangle$	$w_0 + w_1 \sigma_{-1,1}(-2) + w_2 \sigma_{0,1}(-2) + w_3 \sigma_{1,1}(-2) = 18.28$
$\langle -1.25, 4.64 \rangle$	$w_0 + w_1 \sigma_{-1,1}(-1.25) + w_2 \sigma_{0,1}(-1.25) + w_3 \sigma_{1,1}(-1.25) = 4.64$
$\langle -0.5, 0.16 \rangle$	$w_0 + w_1 \sigma_{-1,1}(-0.5) + w_2 \sigma_{0,1}(-0.5) + w_3 \sigma_{1,1}(-0.5) = 0.16$
$\langle 0.25, -0.63 \rangle$	…
$\langle 1, -1.95 \rangle$	…
$\langle 1.75, -9.56 \rangle$	…

$\Phi_{3,2}$

For each point we have an equation, and all together form a linear system.

I am going to represent this system in matrix form, where each element of the matrix of coefficients Φ is the value of a basis function (in our case one of the three sigmoids) on one of the points.

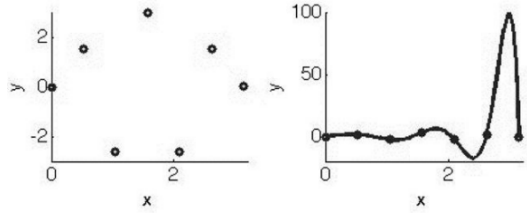
In matrix form

UNIVERSITY OF LEEDS

$$\begin{bmatrix} \Phi_{1,1} & \Phi_{1,2} & \cdots & \Phi_{1,M+1} \\ \Phi_{2,1} & \Phi_{2,2} & \cdots & \Phi_{2,M+1} \\ \vdots & \vdots & \vdots & \vdots \\ \Phi_{N,1} & \Phi_{N,2} & \cdots & \Phi_{N,M+1} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}$$

or: $\boldsymbol{\Phi w = t}$

If $\boldsymbol{\Phi}$ square and nonsingular: $\boldsymbol{w = \Phi^{-1} t}$



No error, the fitted function goes through every point

So, the element in position i,j in this matrix corresponds to the value of the j-th basis function on the i-th point.

If Φ was a square matrix, which means that we'd have exactly as many unknowns as equations we could solve the system exactly, and obtain the parameter vector w that best approximates the function, since our model would pass exactly through each point.

This could be done by inverting the matrix Φ (even though in pratice there are more computationally efficient methods).

Exact solution

UNIVERSITY OF LEEDS

If we only had 4 points, we could find an exact solution, that is a function that goes through the points:

$$\boldsymbol{w} = \boldsymbol{\Phi}^{-1}\boldsymbol{t}$$

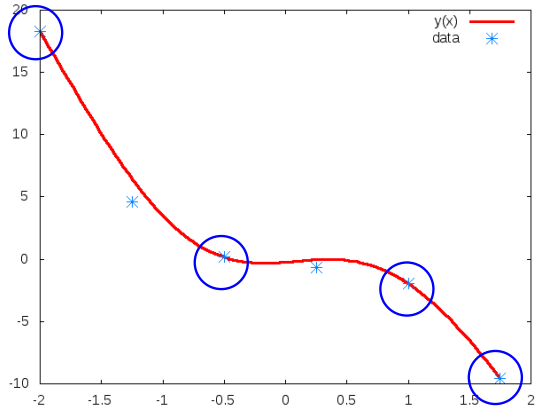
$\langle -2, 18.28 \rangle$

$\langle -0.5, 0.16 \rangle$

$\langle 1, -1.95 \rangle$

$\langle 1.75, -9.56 \rangle$

$$\vec{w} = \langle -34.99, 184.249, -312.42, 209.20 \rangle$$



To apply what I just described, here I have chosen 4 points from my original dataset (since my model has 4 parameters), and solved the linear system of equations exactly.

This is the resulting function, which, as you can see, goes through all 4 points.

Overdetermined vector

UNIVERSITY OF LEEDS

If vector w overdetermined (more equations than variables) → no exact solution available.

Then (guess what), define an error and minimize.

You would like:

$$\boldsymbol{\phi}_i^T(\boldsymbol{x}_i)\boldsymbol{w} = t_i \quad \forall\, \boldsymbol{x}_i$$

So a good error seems to be the difference between the left and right-end side of that equation:

$$E = \frac{1}{2}\sum_i^N \left(\boldsymbol{\phi}_i^T \boldsymbol{w} - t_i\right)^2$$

In general, however, we have more points than parameters.

So our solution cannot go through each point exactly, we have to settle for some amount of error.

The error is the difference between the left and right side of the equation for each point. We want to minimise this difference globally over the whole dataset.

So we minimise the total error: the square of the norm of the difference between the left and right side of the equation for each point, summed over all the points.

Sum-of-squares error

UNIVERSITY OF LEEDS

Let's ignore the vector notation for a minute:

$$E = \frac{1}{2}\sum_{i=1}^{N}(y_i - t_i)^2 = \frac{1}{2}\sum_{i=1}^{N}(\phi_i w - t_i)^2$$

What's the gradient?

$$\nabla E = \sum_{i=1}^{N}\phi_i(\phi_i w - t_i)$$

Sum-of-squares error

UNIVERSITY OF LEEDS

Reintroducing vectors:

$$E = \frac{1}{2}\sum_{i=1}^{N}(y_i - t_i)^2 = \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{\phi}_i^T \boldsymbol{w} - t_i)^2 = \frac{1}{2}\|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{t}\|^2$$

$$\nabla E = \sum_{i=1}^{N}(\boldsymbol{\phi}_i^T \boldsymbol{w} - t_i)\boldsymbol{\phi}_i = \boldsymbol{\Phi}^T(\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{t})$$

You'll remember that the norm of a vector is the square root of the some of each element squared. Therefore, if you look at the second term in the first equation, you can see that it has the same form as the a squared norm (because the square root is gone).

The matrix Φ has in each row the value of the basis function on the corresponding input. So, it has one column per basis function, and one row per input point.

Its gradient in matrix notation has the form below. You can see that it is the same as the previous slide, but with vectors.

Least squares solution

UNIVERSITY OF LEEDS

$$\Phi^T(\Phi w - t) = 0 \qquad \Rightarrow \Phi^T \Phi w = \Phi^T t$$

Necessary condition for a vector w to be a minimum

$\Phi^T \Phi$ Is a square matrix.

If it is also non singular, we can invert it and solve the equation above:

$$w = (\Phi^T \Phi)^{-1} \Phi^T t$$

where $\Phi_p = (\Phi^T \Phi)^{-1} \Phi^T$ is the *pseudoinverse* of Φ

Since the error function is quadratic in the parameters, this is a convex function with a single minimum.

We can compute the minimum in closed form, by solving for when the gradient is 0.

We obtain a solution through a matrix called the *pseudoinverse*, since it plays the same role as the inverse of Φ when Φ is square.

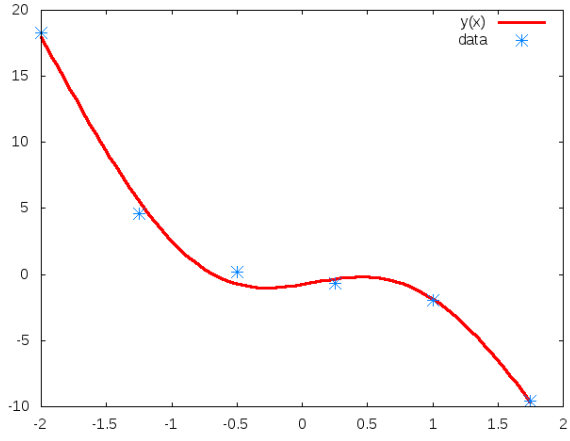
This is the Least Squares solution, and provides a value for the parameters which minimises the total (or average, which is the same) error.

Least-squares solution

UNIVERSITY OF LEEDS

$$\boldsymbol{w} = \left(\boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{t}$$

$$\boldsymbol{w} = \langle -36.53, 198.30, -339.45, 225.09 \rangle$$



In our example, this is the least-squares solution.

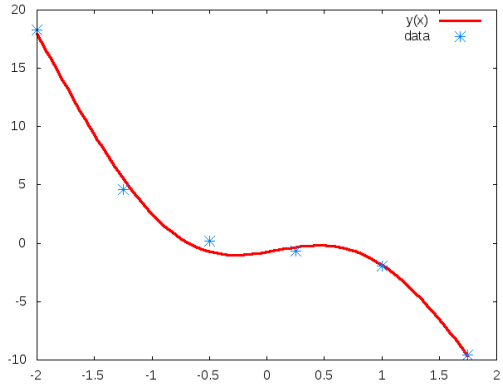
You can see that our function does not go trough every point, but very close to each one.

The sum of the distances between each point and the function is minimised (on average).

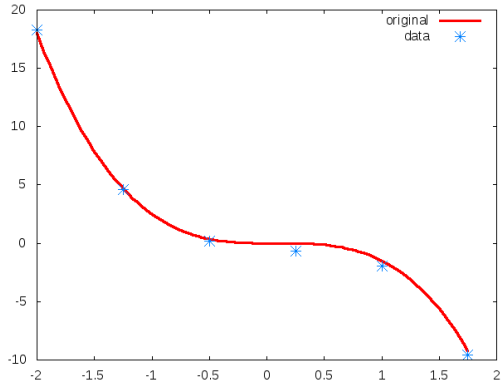
Comparison

UNIVERSITY OF LEEDS

Least-squares



Original function



$$f(x) = 0.5x^2 - 2x^3$$

On the left-hand side is our learned approximation, and on the right-hand side the function I actually used to generate the dataset.

I would say that they are pretty close!

So we used three sigmoids to approximate a function that has nothing to do with sigmoids, through the best possible choice of weights (in terms of total error).

Exercise

UNIVERSITY OF LEEDS

Given the dataset: {(0,0), (0,1), (1,0)}, find the least-squares solution for the parameters in the regression of the function: $y = w_1 + w_2 x^2$

1) Evaluate the bases on the points: $\boldsymbol{\Phi} = ?$

2) Compute: $\boldsymbol{\Phi}^T \boldsymbol{\Phi} = ?$

3) Invert: $\left(\boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} = ?$

4) Compute the pseudo-inverse: $\boldsymbol{\Phi}_p = \left(\boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T = ?$

5) Compute w! $\boldsymbol{w} = \boldsymbol{\Phi}_p \boldsymbol{t} = ?$

Exercise

UNIVERSITY OF LEEDS

Given the dataset: {(0,0), (0,1), (1,0)}, find the least-squares solution for the parameters in the regression of the function: $y = w_1 + w_2 x^2$

1) Evaluate the bases on the points: $\mathbf{\Phi} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$

Exercise

UNIVERSITY OF LEEDS

Given the dataset: {(0,0), (0,1), (1,0)}, find the least-squares solution for the parameters in the regression of the function: $y = w_1 + w_2 x^2$

1) Evaluate the bases on the points: $\mathbf{\Phi} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$

2) Compute: $\mathbf{\Phi}^T \mathbf{\Phi} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 1 & 1 \end{bmatrix}$

Exercise

UNIVERSITY OF LEEDS

Given the dataset: {(0,0), (0,1), (1,0)}, find the least-squares solution for the parameters in the regression of the function: $y = w_1 + w_2 x^2$

1) Evaluate the bases on the points: $\boldsymbol{\Phi} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$

2) Compute: $\boldsymbol{\Phi}^T \boldsymbol{\Phi} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 1 & 1 \end{bmatrix}$

3) Invert: $\left(\boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} = \begin{bmatrix} 0.5 & -0.5 \\ -0.5 & 1.5 \end{bmatrix}$

Exercise

UNIVERSITY OF LEEDS

Given the dataset: {(0,0), (0,1), (1,0)}, find the least-squares solution for the parameters in the regression of the function: $y = w_1 + w_2 x^2$

1) Evaluate the bases on the points: $\boldsymbol{\Phi} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$

2) Compute: $\boldsymbol{\Phi}^T \boldsymbol{\Phi} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 1 & 1 \end{bmatrix}$

3) Invert: $\left(\boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} = \begin{bmatrix} 0.5 & -0.5 \\ -0.5 & 1.5 \end{bmatrix}$

4) Compute the pseudo-inverse: $\boldsymbol{\Phi}_p = \left(\boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T = \begin{bmatrix} 0.5 & 0.5 & 0 \\ -0.5 & -0.5 & 1 \end{bmatrix}$

Exercise

UNIVERSITY OF LEEDS

Given the dataset: {(0,0), (0,1), (1,0)}, find the least-squares solution for the parameters in the regression of the function: $y = w_1 + w_2 x^2$

1) Evaluate the bases on the points: $\boldsymbol{\Phi} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$

2) Compute: $\boldsymbol{\Phi}^T \boldsymbol{\Phi} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 1 & 1 \end{bmatrix}$

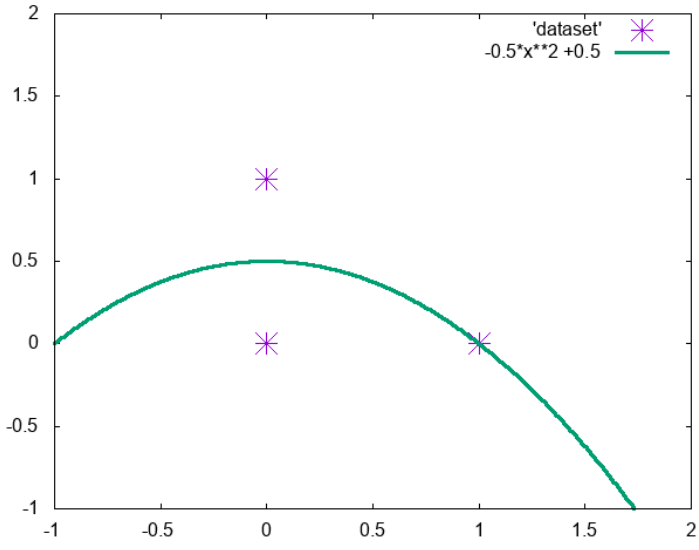
3) Invert: $\left(\boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} = \begin{bmatrix} 0.5 & -0.5 \\ -0.5 & 1.5 \end{bmatrix}$

4) Compute the pseudo-inverse: $\boldsymbol{\Phi}_p = \left(\boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T = \begin{bmatrix} 0.5 & 0.5 & 0 \\ -0.5 & -0.5 & 1 \end{bmatrix}$

5) Compute w! $\boldsymbol{w} = \boldsymbol{\Phi}_p \boldsymbol{t} = \begin{bmatrix} 0.5 & 0.5 & 0 \\ -0.5 & -0.5 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix}$

Result

UNIVERSITY OF LEEDS



The function we computed is a parabola.

Out of all possible parabolas with 2 parameters (this one is missing the term x), it minimises the total or average error over all the points, since it passes through the point <1,0> and exactly in the middle of the two points <0,0> and <0,1>.

You can note, again, that despite the function is linear in the parameters, the learned function is not liner in the input.

Sequential learning

UNIVERSITY OF LEEDS

The closed-form solution requires to process the whole data set.

If $E = \sum_n E_n$

As with neural networks, it is also possible to consider one point at a time

$$w^{(t+1)} = w^{(t)} - \eta \nabla E_n$$

For a least-squares problem:

$$w^{(t+1)} = w^{(t)} - \eta \boldsymbol{\phi}_n^T (\boldsymbol{\phi}_n^T \boldsymbol{w}^{(t)} - \boldsymbol{t}_n)$$

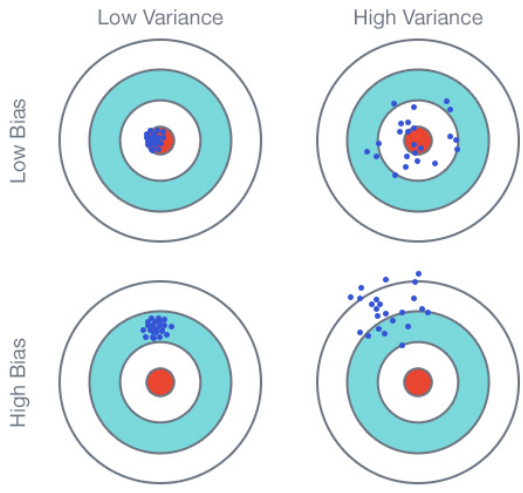
known as the Least-mean-squares (LMS) algorithm

The pseudoinverse is computationally expensive to compute for large datasets, so it is possible to minimise the error through stochastic gradient descent, as we did for neural networks.

In this case you can use mini-batches, and also gain the advantage of learning online, as more points are added to the dataset.

Bias and Variance

UNIVERSITY OF LEEDS



For both classification and regression we saw how the number of parameters and the structure chosen for learning determines the ability to approximate the data.

We discussed how representation power and overfitting are two conflicting phenomena.

Now we try to characterise better how a model aligns with the data.

We consider how a model changes if we take a different training set from the same data. For instance, like when doing cross validation.

We look at two properties: the bias and the variance. The bias is the average error over the whole training set. The variance is the average *distance* between the errors across all detasets.

So, imagine we do 10-fold cross validation, we get 10 values for the error. Their average is the bias of the model we are using for learning. The amount of variation between the 10 values of error is the variance.

The bias captures the ability of the model to approximate the underline properties of the data (the class or the function that generated the data).

The variance captures the dependency of the model on the particular training set used.

Ideally we want both at zero. We are going to see that this is not possible.

Bias-Variance decomposition

UNIVERSITY OF LEEDS

Underlying function plus noise: $f(\vec{x})+\epsilon$ Generates different datasets D

Simplified notation: $\hat{y} \equiv y(\vec{x}; D)$ $f \equiv f(\vec{x})$

Expected value of the error with respect to datasets D (I excluded noise from these calculations, the book has a version with noise):

$$E_D[(\hat{y}-f)^2] = E_D[(\hat{y}-E_D[\hat{y}]+E_D[\hat{y}]-f)^2]$$

$$= E_D[(\hat{y}-E_D[\hat{y}])^2+2(\hat{y}-E_D[\hat{y}])(E_D[\hat{y}]-f)+(E_D[\hat{y}]-f)^2]$$

$$= E_D[(\hat{y}-E_D[\hat{y}])^2+2E_D[(\hat{y}-E_D[\hat{y}])(E_D[\hat{y}]-f)]+E_D[(E_D[\hat{y}]-f)^2]$$

$$2E_D[(\hat{y}-E_D[\hat{y}])](E_D[\hat{y}]-f)$$

$$E_D[(\hat{y}-E_D[\hat{y}])] = E_D[\hat{y}]-E_D[\hat{y}] = 0$$

Let's consider a regression problem, in which we use a model y(x;D) to approximate a function f(x).

We can define the expected value of the error with respect to the training set D.

Bias-Variance decomposition

UNIVERSITY OF LEEDS

$$= E_D[(\hat{y} - E_D[\hat{y}])^2] + 2E_D[(\hat{y} - E_D[\hat{y}])(E_D[\hat{y}] - f)] + E_D[(E_D[\hat{y}] - f)^2]$$

$$= E_D[(\hat{y} - E_D[\hat{y}])^2] + E_D[(E_D[\hat{y}] - f)^2]$$

Variance

Bias^2

Recall that the variance of a random variable is: $var(X) = E[(X - E[X])^2]$

Not limited to regression with mean squared error: general phenomenon

After a few calculations and rearrangements, which I have done for you just in case, we see that the error decomposes exactly into two terms: the bias and the variance.

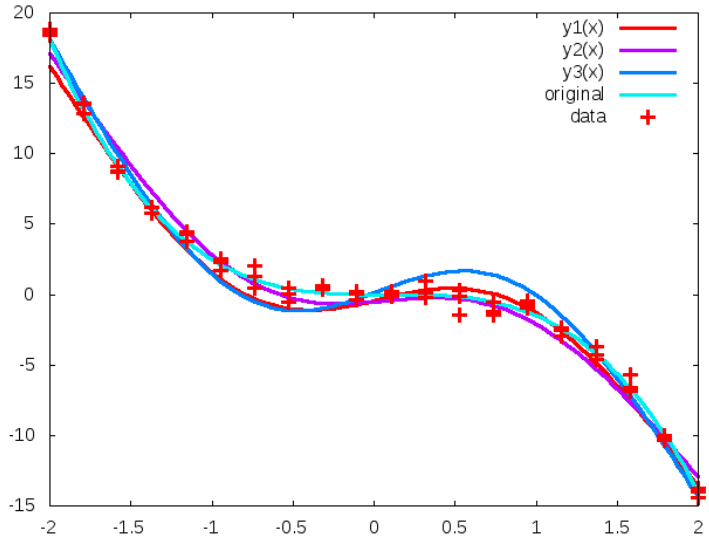
Each model is, in general, only an approximation, so the total error will not be zero. This error is distributed between bias and variance.

Different models strike different balances between the two, and this is what we want to establish with cross-validation.

Overfitting is a consequence of high variance, because the classifier or regressor is learning the noise in the data, therefore by changing the dataset we may have very different models.

Bias-Variance

UNIVERSITY OF LEEDS



Here I generate more points from the same function from the regression example, and added noise, so you also see different values for the same x.

By learning on different subsets of the points we learn different models (in red, magenta and blue).

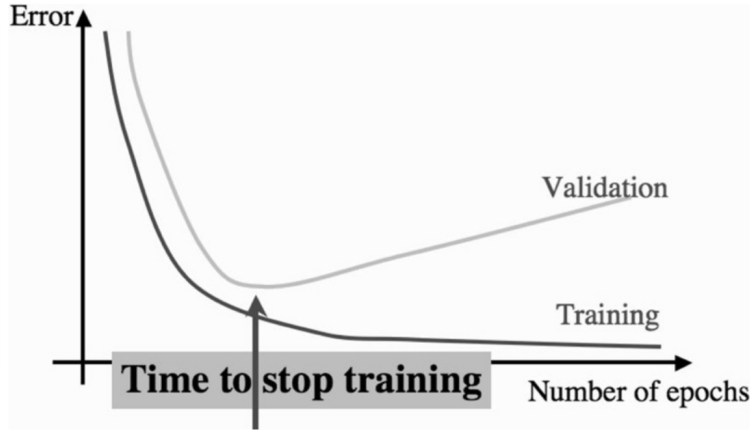
The distance between the average of the learned functions and the real function is the bias.

The average distance between the learned functions is the variance.

We can reduce the bias by adding parameters, but this will cause the variance to increase, unless we have enough data points to support the more complex model.

On training and validation error

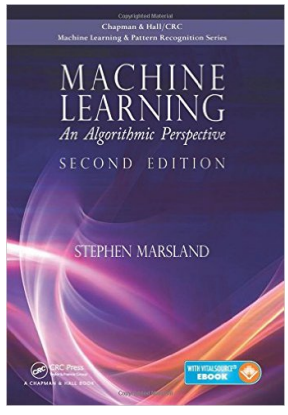
UNIVERSITY OF LEEDS



Overfitting can be interpreted as an effect of the bias-variance decomposition.

By continuing the training we can reduce the bias, that is the error, but at the cost of the variance. Therefore, when we use a different set, such as the validation set, we are likely to see a large variation in the error!

UNIVERSITY OF LEEDS



Chapter 2.5 and 3.5