UNIVERSITY OF BERGEN

INFO 284 - Machine Learning

Linear Regression

Bjørnar Tessem

UNIVERSITY OF BERGEN

Supervised learning reminder

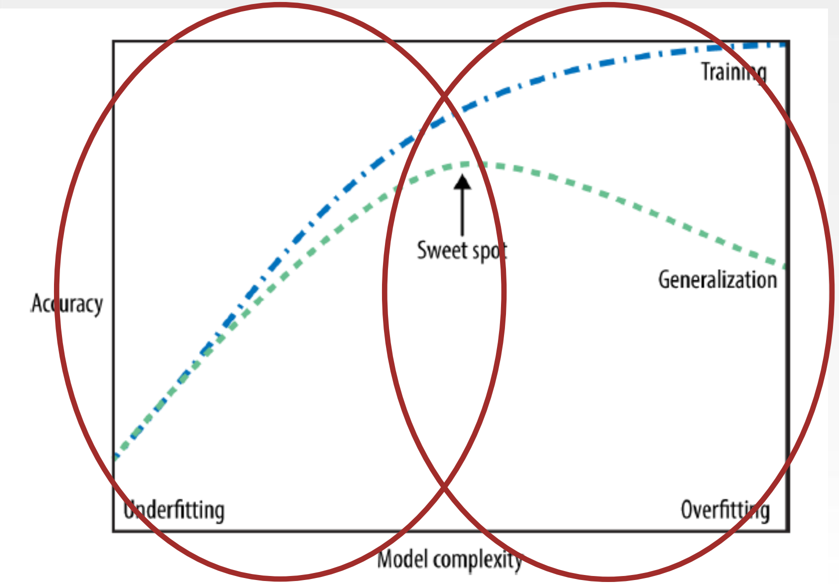
- Supervised learning involves observing several examples of a random vector **x** and an associated value or vector **y**, then learning to predict y from x, often by estimating p(**y**|**x**).

- Given a training set of N example input-output pairs $(x_1,y_1), (x_2,y_2),\ldots,(x_N,y_N)$ where each y_j was generated by an unknown function $y=f(x)$, discover a function h that approximates the true function f.

hypothesis

Capacity

Capacity of a hypothesis =
the amount of functions it is able to model.



- Control Overfitting/underfitting
  - Control the **capacity** of the hypothesis

- Too low capacity - struggle to fit the training set.

- Too high capacity - learn features that are specific to the training set

Vapnik-Chervonenki

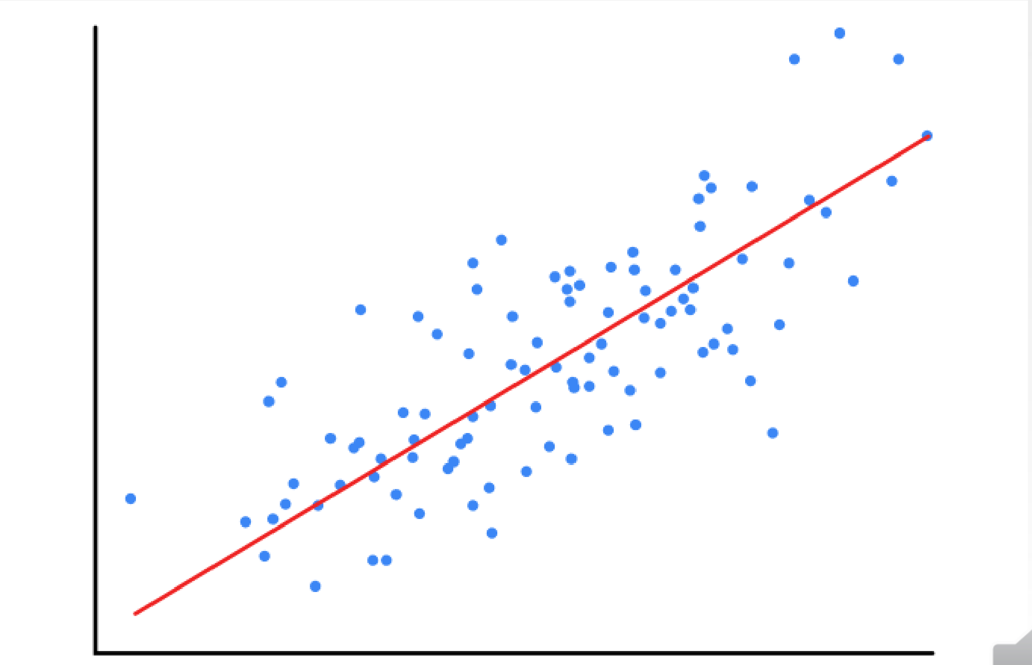
- Control capacity → Choose hypothesis space.
  - Example: only linear functions can be used as hypothesis.

- Statistical learning theory
  - measure capacity of hypothesis space

- Vapnik-Chervonenkis dimension
  - Capacity of a binary classifier.
  - = Cardinality of the largest set of points the algorithm can shatter
  - Linear regression – VC = d+1

Linear regression

Find the line that fits the data best



Linear Models for Regression

- Linear models for regression are regression models for which the hypothesis is
    - **a line** when one feature is used.
    - **a plane** when two features are used.
    - **a hyper-plane** when more features are used

- Linear models for regression can only be used when the features are real numbers!

Linear Regression

- The hypothesis is a **linear function** of the input features.

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \cdots + w[p] * x[p] + b$$

- Learning = trying to find the parameters w[i] and b

Linear Models for Regression

- There are many models/algorithms for linear regression depending on:
  - how the parameters w and b are learned
  - how model/hypothesis complexity can be controlled
- Here: ordinary least squares, ridge regression, lasso

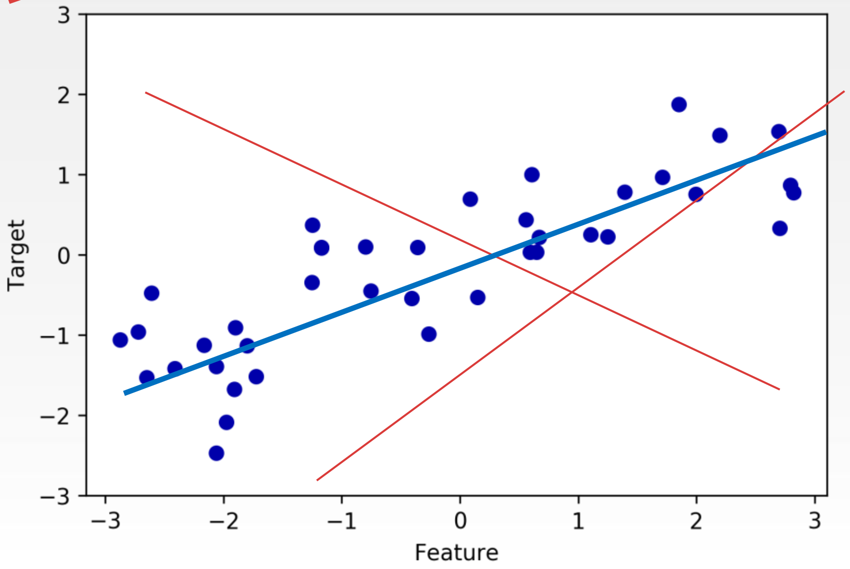
Univariate linear regression

$$\hat{y} = \boxed{w[0]} * x[0] + \boxed{b}$$

offset
Intercept
bias

slope
parameters

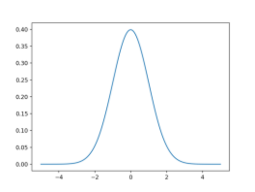
weights
coefficients



Ordinary Least Squares

$$\hat{y} = w[0] * x[0] + b$$

Gaussian (Normal) distribution:



- Gauss showed that if the y values have **normally distributed noise**, then the most likely values of w[0] and b are obtained by minimising the sum of the square of errors.

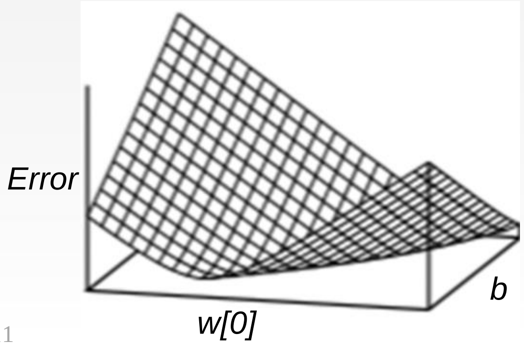
$$\text{Error}(\hat{y}) = \sum_{i=1}^{N} (y_i - (w[0] * x_i[0] + b))^2$$

- We are looking for w[0] and b that make $\text{Error}(\hat{y})$ minimal

Ordinary Least Squares

$$\text{Error}(\hat{y}) = \sum_{i=1}^{N} (y_i - (w[0] * x_i[0] + b))^2$$

- We are looking for w[0] and b that make $\text{Error}(\hat{y})$ minimal.
  Can be found by solving the above as a differential equation



- The problem has a unique solution
  - the «bottom» of the surface

The solution

$$w_0 = \frac{N(\sum x_j y_j) - (\sum x_j)(\sum y_j)}{N(\sum x_j^2) - (\sum x_j)^2}$$

$$b = \frac{\sum y_j - w_0(\sum x_j)}{N}$$

Coefficient of determination R^2

R^2 =evaluates how good a regression algorithm performs

= 1.0: perfect prediction
predicts average → 0.0
Negative: even worse

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

average of actual target values

actual target values

$$R^2 = 1 - \frac{u}{v}$$

residual sum of squares

$$u = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

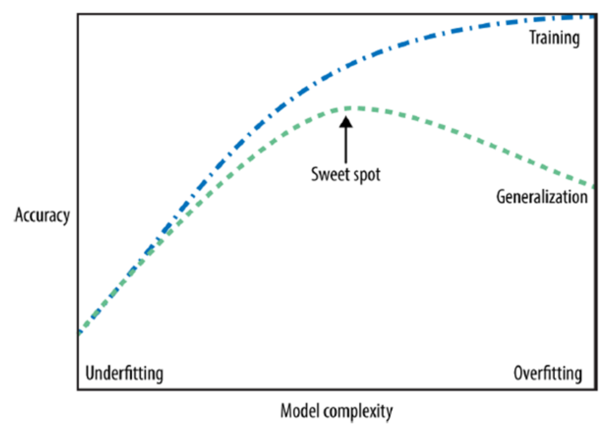
predicted target values

total sum of squares

$$v = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

R^2 and overfitting

- Few data and/or many features will increase chance of overfitting
  - R^2 close to 1 for training set, R^2 significantly smaller for test set → Overfitting
  - R^2 similar for test and training set → good model



Ridge Regression

- Ridge regression is also a linear regression algorithm.
- In addition to fitting to a line we also try to make the coefficients to be as small as possible

$$Error(\hat{y}) = \sum_{i=1}^{N}(y_i - \underbrace{\boxed{(w[0]) * x_i[0] + b)}}_{\hat{y}_i})^2 + \alpha * \sqrt{w[0]^2}$$

L2 Regularisation

Ridge regularisation

- The Ridge model makes a trade-off between the simplicity of the model and its performance on the training set
- Fine-tuning with the alpha parameter
- Large alpha → coefficients move towards zero
    - decreases the performance on the training set
    - might help generalisation
- Alpha close to 0.0
    - similar results to OLS

On Norms

- In mathematics a **norm** is a function that assigns a positive *number* to each vector in a vector space.
- The norm of a mathematical object is a quantity that in some (possibly abstract) sense describes the length, size, or extent of the object.

Various Norms

- The **L2** norm

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

- The **L1** norm is just the sum of the absolute values

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

- The **L0** norm = how many non zero values are there in the vector

Regularisation

- Regularisation adds a penalty on the different parameters of the **hypothesis** to reduce its complexity.

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \cdots + w[p] * x[p] + b$$

LASSO

- Least Absolute Shrinkage and Selection Operator
- Also restricts coefficients to values close to zero but in a different way
  - **L1 regularisation**.
- Consequence
  - some w[i] are set to zero, which is the same as ignoring the x[i] features

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \cdots + w[p] * x[p] + b$$

LASSO

- Ridge regression is usually the choice over Lasso
- When there are a lot of features, and you expect some of them are not needed, then Lasso is a better choice

Types of regularisation

- No regularisation - Ordinary least squares.
  Learning = finding the minima for the function:

$$Error(\hat{y}) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

- L2 regularisation - Ridge penalisation.

Learning = finding the minima for the function:

$$Error_{L2}(\hat{y}) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \alpha * \|\boldsymbol{w}\|_2$$

- L1 regularisation - LASSO penalisation.

Learning = finding the minima for the function:

$$Error_{L1}(\hat{y}) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \alpha * \|\boldsymbol{w}\|_1$$

uib.no