Machine learning fundamentals

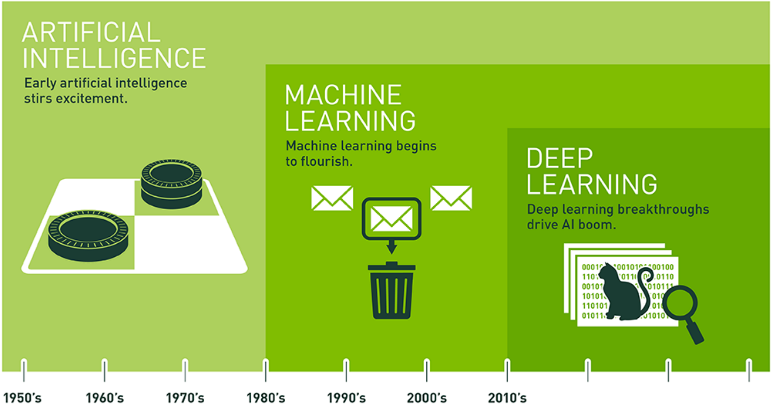
Deep learning course for industry

Mitko Veta

Eindhoven University of Technology
Department of Biomedical Engineering

2020

Historical perspective


ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Figure source: nvidia.com

Course overview

Day 1:

- Theory
  - Machine learning fundamentals
  - From linear models to deep neural networks
  - Convolutional neural networks
- Practice
  - Linear and logistic regression in Keras
  - Fully connected neural networks in Keras
  - Convolutional neural networks in Keras

Course overview

Day 2:

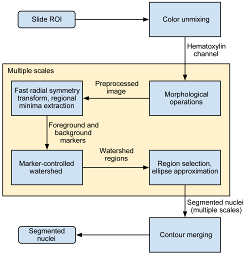
- Theory
  - Experimental methodology for training of ML/DL models
  - Overview of modern neural network architectures
- Practice
  - Image segmentation with U-Net
  - Mini-competition: Segmentation of cardiac MR images

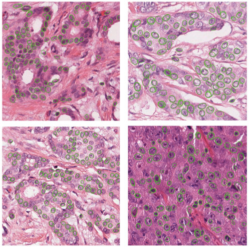
Learning goals

- Define machine learning.
- Introduce the conceptually simple yet practically useful linear model.
- Discuss the central challenge of machine learning: generalisation.

An example from my past work: nuclei area measurement

2010-2011: An image processing pipeline of (mainly) mathematical morphology operators (e.g. the watershed algorithm).


Slide ROI
Color unmixing
Hematoxylin channel
Multiple scales
Fast radial symmetry transform, regional minima extraction
Preprocessed image
Morphological operations
Foreground and background markers
Marker-controlled watershed
Watershed regions
Region selection, ellipse approximation
Segmented nuclei (multiple scales)
Segmented nuclei
Contour merging

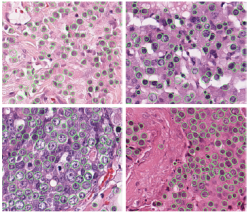


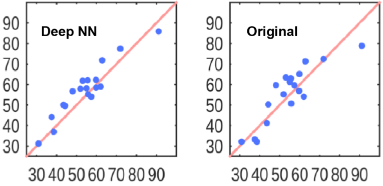
The design and validation of the processing pipeline took the better part of a year.

Figure source: Veta et al. PLOS ONE 2012

An example from my past work: nuclei area measurement

2015: A deep neural network for nuclei area measurement.





Manual vs. automatic nuclear area in µm^2

The the training and validation of the deep neural network model took less than a week.

The results were more accurate than the the original method.

Figure source: Veta et al. MICCAI 2016

An example from my past work: nuclei area measurement

In the first case, I translated the domain knowledge of (medical) experts about nuclei appearance into a series of **manually written rules** that perform nuclei segmentation.

In the second case, I took a dataset of nuclei segmentations and fed it to a (deep) machine learning algorithm that **learned** how to directly measure nuclei size **from the provided examples**.

The central premise of machine learning

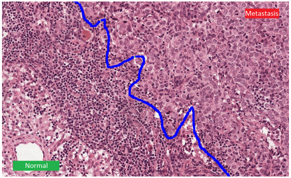
Learn “computer programs” from examples instead of manually writing rules.

The central premise of machine learning

Learn “computer programs” from examples instead of manually writing rules.

Advantage: the same method (e.g. a neural network) can be used to solve a variety of different problems.

Siberian hustky vs. eskimo dog



Normal vs. metastases

Figures source: (left) Szegedy et al. arXiv 2014, (right) camelyon16.grand-challenge.org

The central premise of machine learning

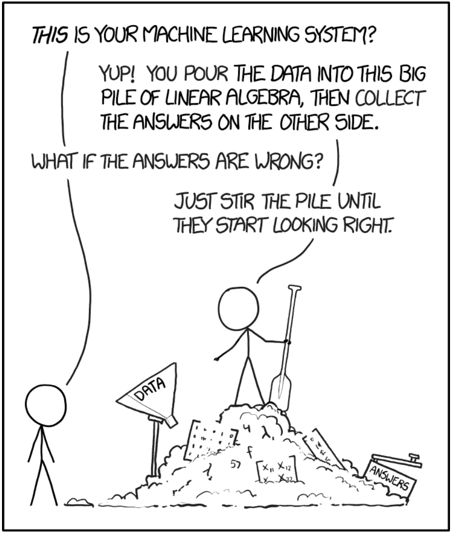

THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

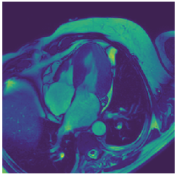
Figure source: xkcd.com

What are the ”examples”?

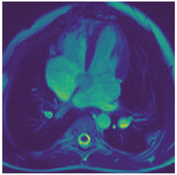
Depends on the particular problem and task.

Dataset: cardiac MRI images.
Task: detect if a specific pathology is present in each image.

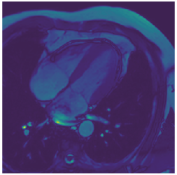
In this case, every image is an example and is associated with a binary target: $0 =$ “healthy”, $1 =$ “diseased” (i.e. we want to classify each image as “healthy” or “diseases”).



diseased



diseased

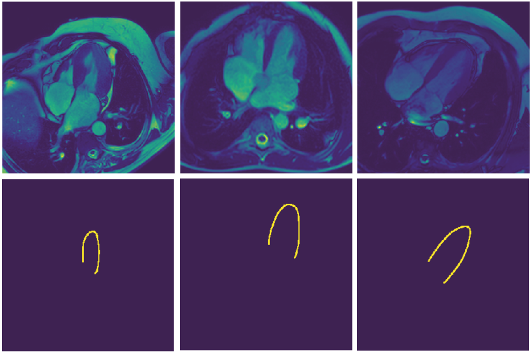


healthy

What are the "examples"?

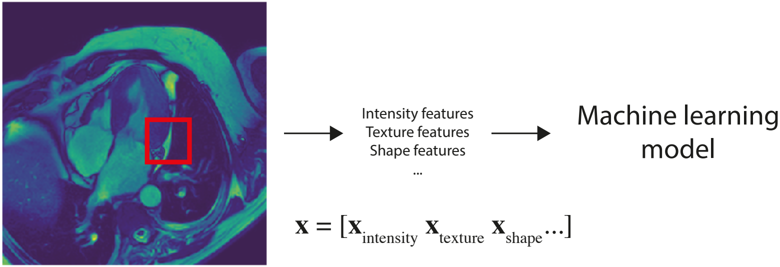
Dataset: cardiac MRI images.
Task: Segment the contours of the left ventricle

In this case, each pixel is an example and is associated with a binary target: 0 = "background", 1 = "contour".



How are the “examples” represented?

Traditionally with feature extraction:



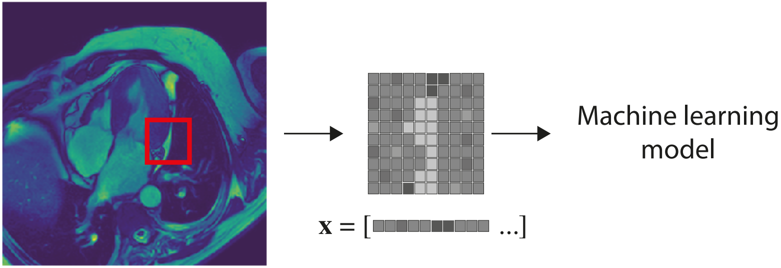
Intensity features
Texture features
Shape features
...

Machine learning model

$$\mathbf{x} = [\mathbf{x}_{\text{intensity}}\ \mathbf{x}_{\text{texture}}\ \mathbf{x}_{\text{shape}} \ldots]$$

How are the “examples” represented?

With raw pixel values (the *de facto* standard for deep learning):



$\mathbf{x} = [\ldots]$

Machine learning model

In summary...

In order to design a machine learning algorithm for a specific task we are given a dataset of examples represented by $\mathbf{x}_i$.

Each example is (optionally) associated with a target y_i.

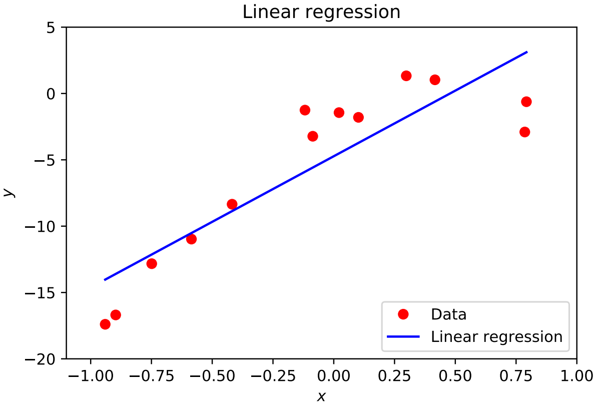
The target can be categorical, such as class membership (e.g. $y_i = \{0, 1\}$), or continuous (e.g. area, volume etc.).

Types of machine learning

- Unsupervised machine learning: given a dataset x_i, find "some interesting properties".
  - Clustering: find groupings of x_i
  - Density estimation: find $p(x_i)$
  - Generative models.
  - ...
- Supervised machine learning: given a training dataset $\{x_i, y_i\}$, predict $\hat{y}_i$ of previously unseen samples.
  - Regression: the target variables y_i are continuous.
  - Classification: the target variables y_i are continuous.
  - ...

A simple machine learning model for regression

The predictions $\hat{y}_i$ are a linear combination of the inputs:



$$\hat{y} = \hat{w}_0 + \sum_{j=1}^{p} x_j \hat{w}_j$$

Linear models are surprisingly useful and common

Fetal weight estimate from ultrasound imaging:

fetal weight $= \hat{w}_0 + \hat{w}_1 \times$femur len. $+ \hat{w}_2 \times$abdominal circ. $+ \hat{w}_3 \times$head circ.

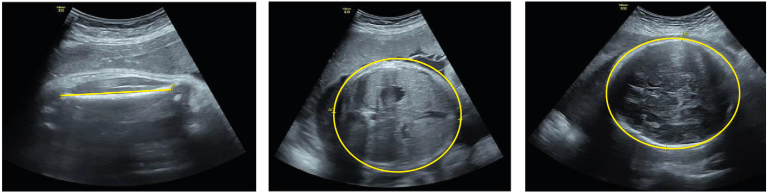


Figure source: my daughter

Linear model

- Input vector $\boldsymbol{x}^T = (x_1, x_2, \ldots, x_p)$.
- Output y predicted using the model

$$\hat{y} = \hat{w}_0 + \textstyle\sum_{j=1}^{p} x_j \hat{w}_j$$

- $\hat{w}_i$ $(0 \leq i \leq p)$ are the parameters of the linear model.

Linear model

- In vector form
$$\hat{y} = \hat{\boldsymbol{w}}^T \boldsymbol{x} = \boldsymbol{x}^T \hat{\boldsymbol{w}}$$
using the fact that the scalar (inner) product of two vectors is a commutative operation.
- We assume that w_0 is in $\boldsymbol{w}$ and 1 is included in $\boldsymbol{x}$.
- $\hat{y}$ is a scalar, but in general can be a k-vector $\hat{\boldsymbol{y}}$, in which case $\boldsymbol{w}$ becomes a $p \times k$ matrix of coefficients.

Linear model fit by least squares

- We need to find coefficients $\hat{w}_i$ which minimise the error estimated with the **residual sum of squares**

  $$\mathrm{RSS}(\boldsymbol{w}) = \sum_{i=1}^{N} (y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2$$

  assuming N input-output pairs (the dataset).
- $\mathrm{RSS}(\boldsymbol{w})$ is a quadratic function.
- A minimum always existsthough not necessarily a unique one.

Linear model fit by least squares

- $\boldsymbol{y} = [y_1, y_2, \ldots, y_N]^T$ is the vector formed from the N output vectors and $\boldsymbol{X}$ is an $N \times p$ matrix

$$\mathrm{RSS}(\boldsymbol{w}) = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

Linear model fit by least squares

- $\boldsymbol{y} = [y_1, y_2, \ldots, y_N]^T$ is the vector formed from the N output vectors and $\boldsymbol{X}$ is an $N \times p$ matrix

$$\mathrm{RSS}(\boldsymbol{w}) = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

- To find the minimum we differentiate with respect to $\boldsymbol{w}$ which gives:

$$(-\boldsymbol{X})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) + (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(-\boldsymbol{X})$$

  using the rule $(\boldsymbol{A}\boldsymbol{B})^T = \boldsymbol{B}^T\boldsymbol{A}^T$ this is equivalent to

$$-2\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

Linear model fit by least squares

- To find the minimum our derivative must be $\mathbf{0}$, hence:

$$\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) = \mathbf{0}$$

$$\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \mathbf{0}$$

$$\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w}$$

- If $\boldsymbol{X}^T\boldsymbol{X}$ is non-singular there exists a unique solution given by

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Question: Why not simply $\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w} = \mathbf{0} \rightarrow \boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} \rightarrow \hat{\boldsymbol{w}} = \boldsymbol{X}^{-1}\boldsymbol{y}$?

Linear model fit by least squares

- For each input $\boldsymbol{x}_i$ there corresponds the fitted output

$$\hat{y}_i = \hat{y}_i(\boldsymbol{x}_i) = \hat{\boldsymbol{w}}^T \boldsymbol{x}_i$$

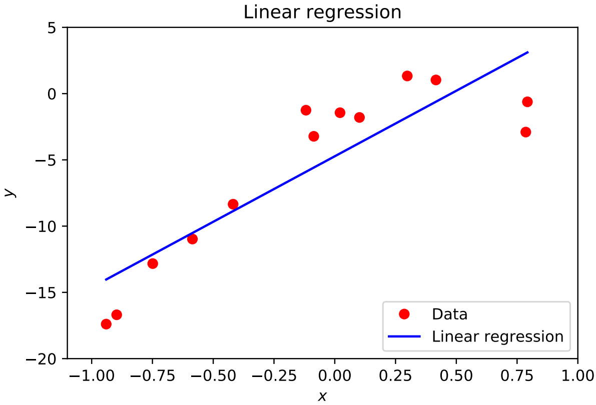
.

- This is called "making a prediction" for $\boldsymbol{x}_i$.
- The entire fitted surface (hyperplane) is fully characterised by the parameter vector $\hat{\boldsymbol{w}}$.
- After fitting the model, we can "discard" the training dataset.

The ML workflow (thus far)

- Collect dataset $\{\mathbf{x}_i, y_i\}$.
- Assume a model for $\hat{y}$.
- Decide on an **error/loss function** that measures the "goodness of fit" of $\hat{y}$ to $\{\mathbf{x}_i, y_i\}$.
- Fit the model to the data with an optimisation procedure (e.g. gradient-based optimisation).

But what if a linear model is not enough?



$$\hat{y} = \hat{w}_0 + \sum_{i=1}^{p} x_i \hat{w}_i$$

$$\hat{y} = \boldsymbol{x}^T \hat{\boldsymbol{w}}$$

Polynomial regression

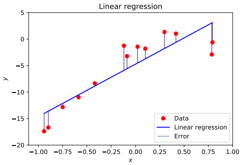
- The linear regression algorithm can be generalised to include all polynomial functions instead of just the linear ones.
- The linear regression model is then just a special case restricted to a polynomial of degree one: $\hat{y} = b + wx$.
- Moving to degree two to we obtain: $\hat{y} = b + w_1x + w_2x^2$.

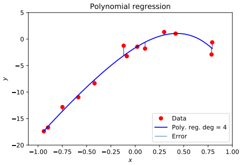
Polynomial regression

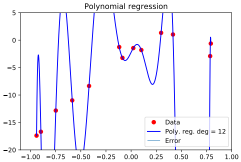
- The linear regression algorithm can be generalised to include all polynomial functions instead of just the linear ones.
- The linear regression model is then just a special case restricted to a polynomial of degree one: $\hat{y} = b + wx$.
- Moving to degree two to we obtain: $\hat{y} = b + w_1 x + w_2 x^2$.
  - **This can be seen as adding a new feature x^2.**
  - In fact, we can generalise this approach to create all sorts of hypothesis spaces, e.g.: $\hat{y} = b + w_1 x + w_2 \sin(x) + w_3 \sqrt{x}$.
- The **output** is still a **linear** function of the parameters, so it can be fitted with least squares.

Polynomial regression

A comparison of a linear, degree-4, and degree-12 polynomials as predictors

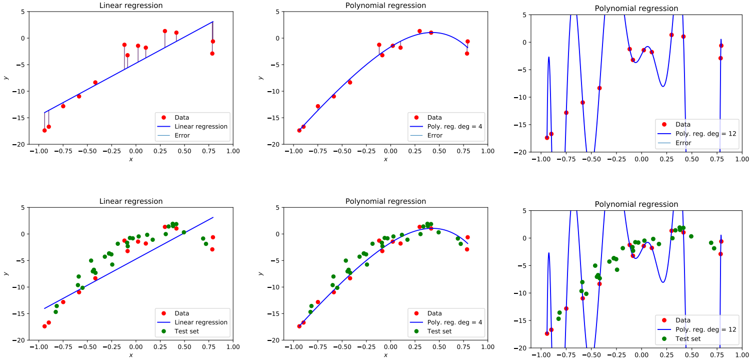






Polynomial regression

A comparison of a linear, degree-4, and degree-12 polynomials as predictors

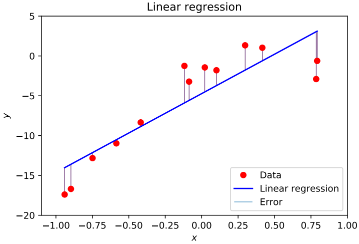


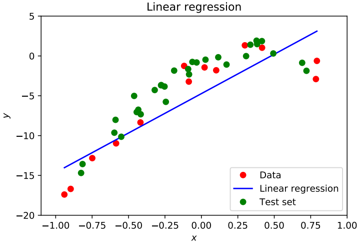
Generalisation

- The central challenge in machine learning is to design an algorithm that will **perform well on new data** (different from the training set data).
- This ability is called **generalisation**.
- **Training error** is the error computed on the training set.

Generalisation

- During the training (learning) we aim at reducing the training error.
- If that is the end goal, we only have an optimisation problem, not a machine learning one.



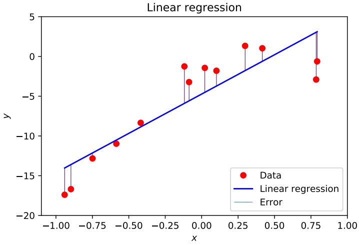


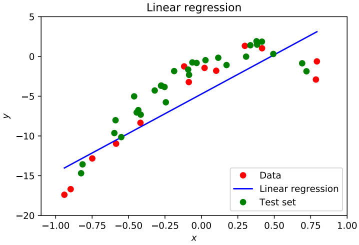
Generalisation error

- **Generalisation error**, also called **test error** is defined as the expected error on new, previously unseen data.
- Unlike in simple optimisation, in machine learning our main goal is to minimise the **generalisation error**.
- Usually the generalisation error is estimated by measuring the performance on a **test data set** which must be **independent** from the training set.

Example: Linear regression

- Previously, we trained the model by minimising the training error
$$\frac{1}{m^{(\text{train})}}\left\|\boldsymbol{X}^{(\text{train})}\hat{\boldsymbol{w}}-\boldsymbol{y}^{(\text{train})}\right\|_2^2$$
- We would like actually to minimise the test error
$$\frac{1}{m^{(\text{test})}}\left\|\boldsymbol{X}^{(\text{test})}\hat{\boldsymbol{w}}-\boldsymbol{y}^{(\text{test})}\right\|_2^2$$





Statistical learning theory

- **Statistical learning theory** provides methods to mathematically reason about the performance on the test set although we can observe only the training set.
- This is possible under some assumptions about the data sets
  - The training and test data are generated by drawing from a probability distribution over data sets. We refer to that as **data-generating process**.
  - **i.i.d. assumptions**
    - Examples in each data sets are **independent** from each other.
    - The training data set and the test data set are **identically distributed**, i.e., drawn from the same probability distribution.

Underfitting and overfitting

- The factor that determines how well a machine algorithm will perform is its ability to
  1. Make the training error small.
  2. Make the difference between the training and test error small.
- These two factors correspond to the two central challenges in machine learning: **underfitting** and **overfitting**.
- Underfitting occurs when the model is not able to produce a sufficiently small training error.
- Overfitting occurs when the gap between the training and test errors is too large.

Model capacity

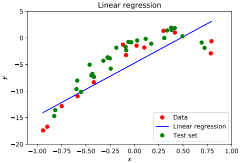
- A **capacity of the model** is its ability to fit a wide variety of functions.
- Low capacity models struggle to fit the training set (underfitting).
- Models with high capacity have danger to overfit the training data (e.g., by “memorising” training samples).

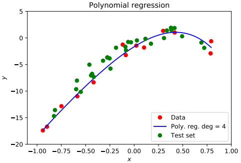
Model capacity

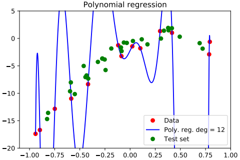
- A **capacity of the model** is its ability to fit a wide variety of functions.
- Low capacity models struggle to fit the training set (underfitting).
- Models with high capacity have danger to overfit the training data (e.g., by "memorising" training samples).
- The capacity can be controlled by choosing its **hypothesis space**, i.e. the set of functions from which the learning algorithm is allowed to select the solution.
- Example: The linear regression algorithm has the set of all linear functions as its hypothesis space.

Underfitting and overfitting in polynomial estimation

- Models with low capacity are not up to the task.
- Models with high-capacity can solve a complex task, but when the capacity is too high for the concrete (training) task there is the danger of overfitting.



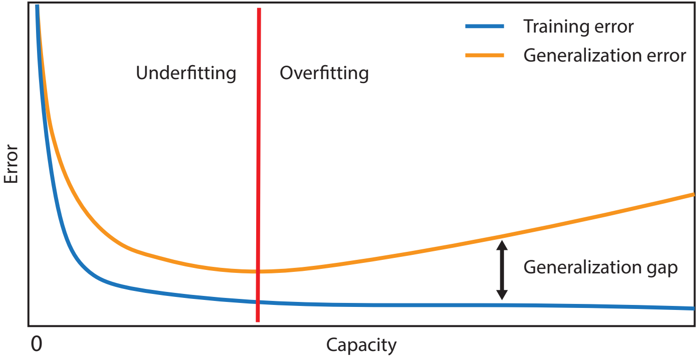




Generalisation and capacity

- Simpler functions generalise more easily, but we still need to choose a sufficiently complex hypothesis (function) to obtain small training error.
- Typically training error decreases with the increase of the model capacity until an (asymptotic) value is reached.
- The generalisation error is U-shaped with the capacity range split in an underfitting and an overfitting zone.

Generalisation and capacity



Training set size

- Training and generalisation error vary as the size of the training data set varies.
- Expected generalisation error never increases as the size of the training set increases.
- Any fixed parametric model will asymptotically approach an error value that exceeds the so called Bayes error.
- It is possible for the model to have optimal capacity and still have a large gap between training and generalisation errors.
- In that case the gap usually can be reduced with increasing the number of training examples.

Training set size

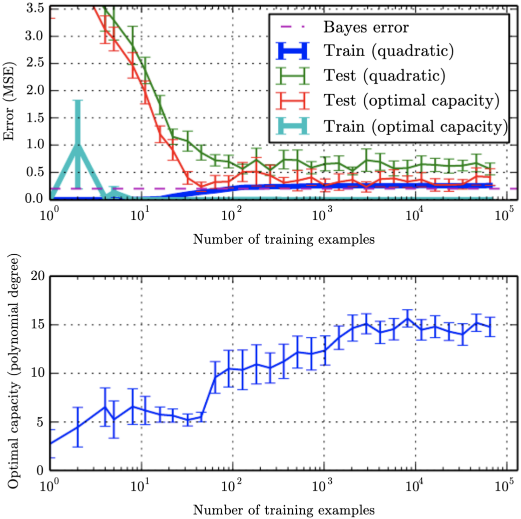


Figure source: deeplearningbook.org

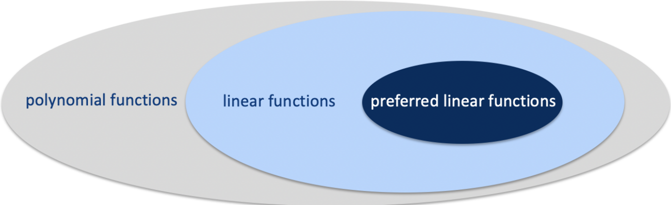
Regularisation

- In addition to increasing and decreasing of the hypothesis space, i.e., the capacity, we can influence the learning algorithm by **giving preference to one solution over another in the hypothesis space**.
- In case both functions are eligible we can define a condition to express preference about one of the functions.
- The less preferred solution is chosen only if it gives significantly better performance with the training data.

Regularisation

- In addition to increasing and decreasing of the hypothesis space, i.e., the capacity, we can influence the learning algorithm by **giving preference to one solution over another in the hypothesis space**.
- In case both functions are eligible we can define a condition to express preference about one of the functions.
- The less preferred solution is chosen only if it gives significantly better performance with the training data.
- E.g., prefer smaller weights **w**:

$$L(\mathbf{w}) = RSS(\mathbf{w}) + \lambda \mathbf{w}^\intercal \mathbf{w}$$



Summary

- Machine learning studies algorithms that learn from examples instead of relying on manually written rules.
- The linear model is conceptually simple but practically useful and can be seen as the basic building block of neural networks.
- The central challenge in machine learning is to find a model that will perform well on new data. This ability is called generalisation.