#DLUPC

Day 1 Lecture 2

Machine Learning

Verónica Vilaplana
veronica.vilaplana@upc.edu

Associate Professor
Universitat Politecnica de Catalunya
Technical University of Catalonia

UPC

Index

- Introduction
  - What is machine learning
  - Types of ML problems

- Supervised learning
  - Learning phases
  - Evaluating a predictor: loss functions
  - Expected risk vs empirical risk
  - Model selection - Capacity
  - Overfitting – undefitting
  - Bias-variance tradeoff
  - Cross validation

Introduction

What is machine learning?

- A scientific field that explores the study and construction of algorithms that can learn from data and produce accurate predictive functions applicable to similar data (may also yield informative descriptive functions of data)

- The key element of machine learning is data
  - Collected from nature or (industrial) processes
  - In many forms and formats, structured, unstructured, clean or noisy
  - Viewed as a list of examples (ideally, many examples of the same nature)

Types of machine learning problems

Based on the information available

- **Supervised learning (and semi-supervised)**: predict a target y from input x
- **Unsupervised learning**: no explicit prediction target y
- **Reinforcement learning**: agent learns by interacting with its environment and observing the results of these interactions. .

Supervised learning

- Training experience: a set of labeled examples of the form $(\mathrm{x}; y)$
- What to learn: a function (parameter θ) to predict y from x $f_\theta(\mathrm{x}) = y$
  - ideally, we would like to minimize error on all possible instances (we only have a limited set)
- Problems are categorized by the **type of output domain**
  - y represents a category or "class" ➡ classification

    binary : $\mathbf{y} \in \{-1, +1\}$ or $\mathbf{y} \in \{0, 1\}$

    multi-class : $\mathbf{y} \in \{1, m\}$ or $\mathbf{y} \in \{0, m-1\}$
  - y is a real-value number ➡ regression

    $\mathbf{y} \in R$ or $\mathbf{y} \in \mathrm{R}^m$
  - y could be more complex (graph, tree, etc.) ➡ structured prediction

predictive modeling

Source: P. Vincent / D. Precup

Example: classification

- Image classification: produce a classifier to map from image (pixels) to the category

Apple

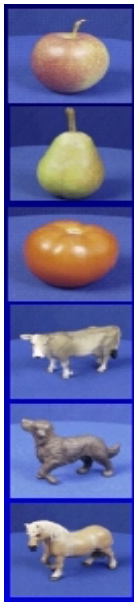
Pear

Tomato

Cow

Dog

Horse

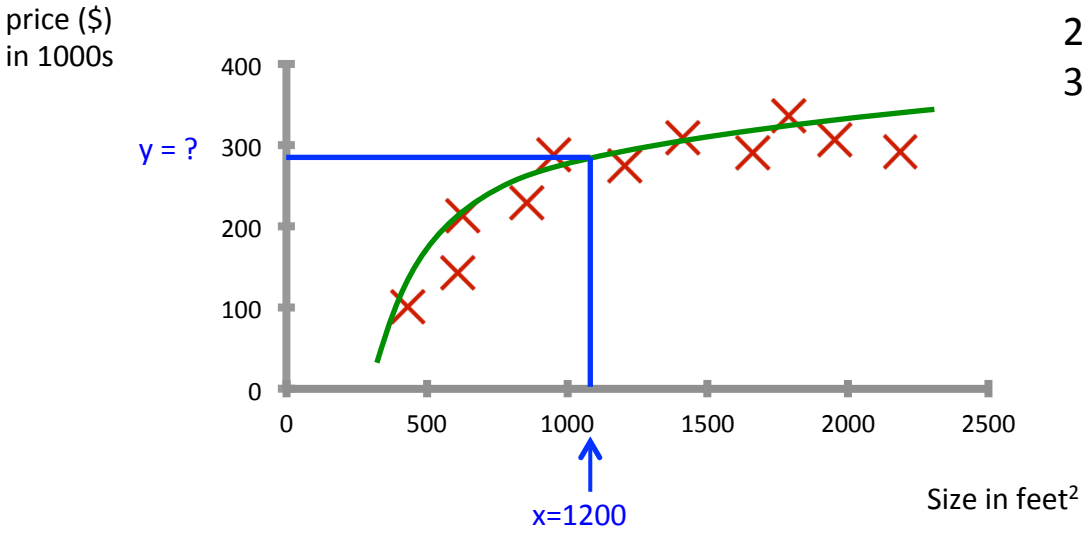


Given: training images and their categories
Color images: $x \in R^{3mm}$ y={apple,….., horse}
Multi-class problem

What are the categories of these test images?

Example: 1D regression

- Housing price prediction: predict a continuous valued output (price)



1. Collect training data
2. Learn a function input → target
3. Use learned function on new inputs

Source: A. Ng

Unsupervised learning

- Training experience: unlabelled data x

- What to learn: interesting associations in the data
  - often there is no single correct answer

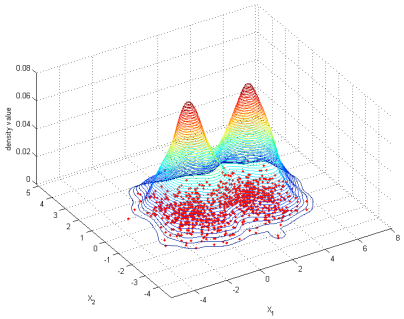
- Model the probability distribution of x
  - ➡ density estimation
- Discover underlying structure in data
  - ➡ clustering
  - ➡ dimensionality reduction
  - ➡ (unsupervised) representation learning

Descriptive modeling

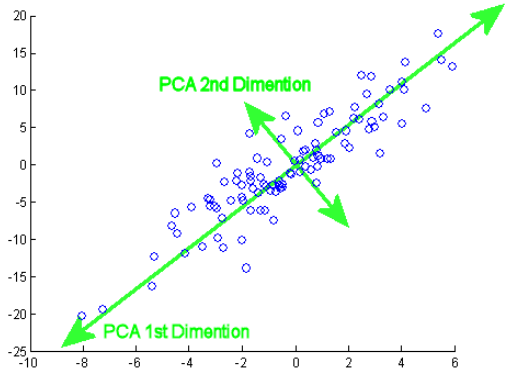
Source: P. Vincent / D. Precup

Examples: density estimation / dim. reduction

- Density estimation
  - Finding a function that approximates the probability density of the data
  - Can be used to detect anomalies



- Dimensionality reduction
  - Finding a lower dimensional representation of the data
  - Useful for compression, visualization, noise reduction



Principal Component Analysis

Example: clustering

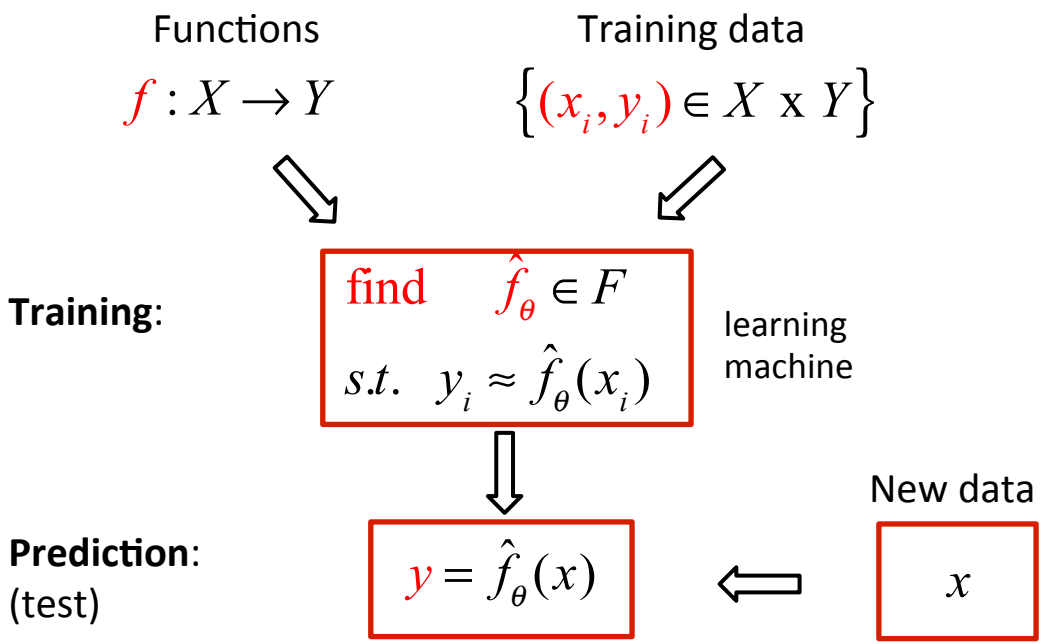
- Custering:
  - Finding a group structure in the data (discover groups of ”similar” data points)
    - Data in one cluster similar to each other
    - Data in different clusters dissimilar

Cluster faces based on identity

[Guillaumin, Verbeek, Schmid, ICCV 2009]

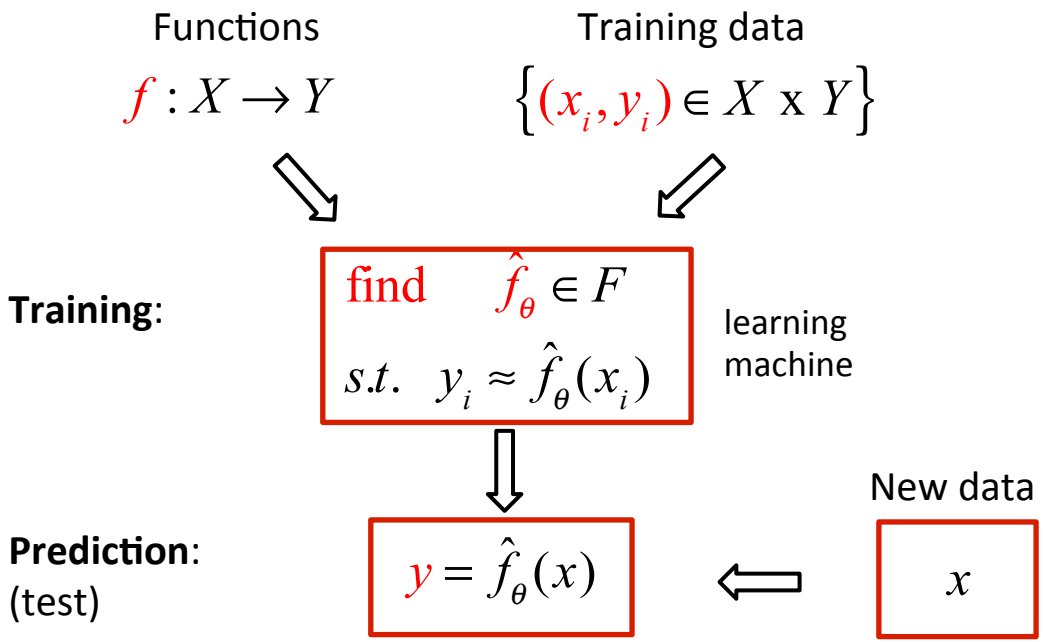
Supervised learning

Learning phases



Key: not to learn perfectly (memorize) but **generalize** well on new (unseen) examples

Learning phases



Which function family F?
often a parameterized family

How to evaluate the quality of f?
using a cost (loss) function L

How to search for the 'best' f ?
optimization of parameters to minimize the loss over the training set

Evaluating a predictor f(x)

The performance of a predictor is often evaluated using several different evaluation metrics:

- Evaluations of true quantities of interest (# lives saved, $ saved,…) when using the predictor inside a more complicated system
- Standard evaluation metrics in a specific field
- Misclassification error rate for a classifier (or precision, recall, F-score,…)
- The loss actually being optimized by the ML algorithm
  - often different from all above

Source: P. Vincent

Standard loss-functions

- For a density estimation task
  - negative log likelihood loss

$$f : R^d \rightarrow R^+ \quad \text{a probability mass or density function}$$
$$L(f(x)) = -\log f(x)$$

- For a regression task
  - squared error loss

$$f : R^d \rightarrow R$$
$$L(f(x)\ , y) = (f(x) - y)^2$$

- For a classification task
  - 0-1 loss or misclassification error loss

$$f : R^d \rightarrow \{0,1,\ldots,m-1\}$$
$$L(f(x)\ , y) = I_{(f(x)\neq y)}$$
$$I_{(f(x)\neq y)} = \begin{cases} 1 & f(x) \neq y \\ 0 & f(x) = y \end{cases}$$

Surrogate loss

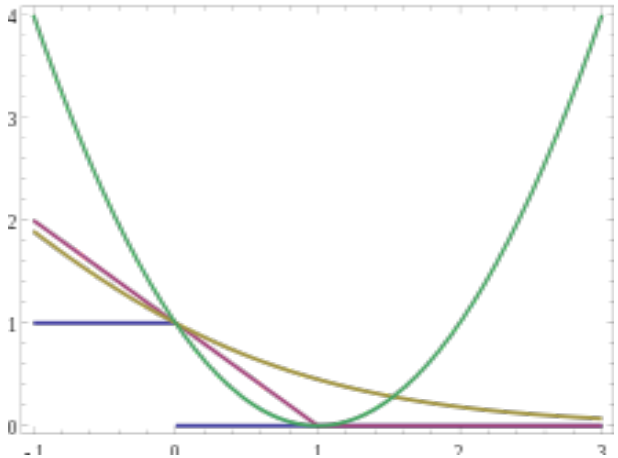
- Often minimizing the real loss is intractable
  - e.g. 0-1 loss $L(f(x),y) = I_{(f(x)\neq y)}$

- We minimize a surrogate loss instead
  - e.g. for the 0-1 loss

hinge $L(f(x),y) = \max(0,1-yf(x))$

square $L(f(x),y) = (1-yf(x))^2$

logistic $L(f(x),y) = \log(1+e^{-yf(x)})$



0-1 loss (blue) and surrogate losses (square, hinge, logistic)

Expected risk vs empirical risk

- Samples (x,y) are supposed drawn i.i.d from an unknown true distribution p_{data}

- Expected risk = generalization error
  - how poorly we will do on average on all examples from the unknown distribution

$$J(f) = \mathrm{E}_{(x,y)\sim p_{data}} L(f(x), y)$$

- Empirical risk = average loss on a finite dataset
  - how poorly we do on average on a finite dataset (distribution $\hat{p}_{data}$)

$$\hat{J}(f, D) = \mathrm{E}_{(x,y)\sim \hat{p}_{data}} L(f(x), y) = \frac{1}{|D|} \sum_{(x^{(i)}, y^{(i)})\in D} L(f(x^{(i)}), y^{(i)})$$

$|D|$ is the number of examples in D

Training and testing phases

How to find the best predictor?

The expected risk cannot be computed: empirical risk minimization principle

- **Training:** find predictor that minimizes the empirical risk on a training set D_{train}

$$\hat{f}(D_{train}) = \arg\min_{f \in F} \hat{J}(f, D_{train})$$

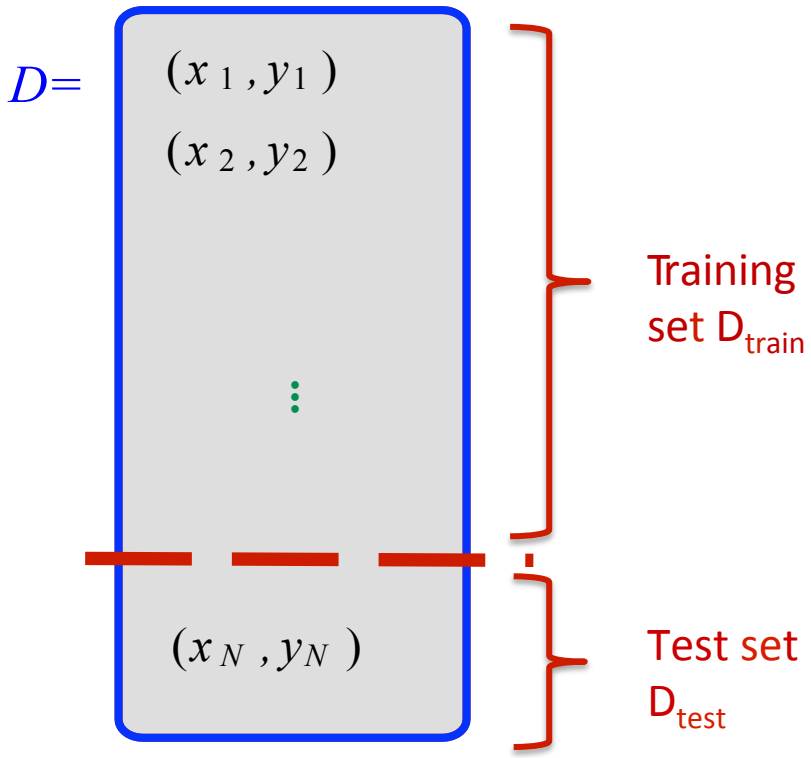
How to evaluate the generalization error of the predictor $\hat{f}(D_{train})$?

- **Testing:** estimate generalization error by the average error on a separate test set D_{test} (never used for training)

$$J(\hat{f}(D_{train})) \approx \hat{J}(\hat{f}(D_{train}), D_{test})$$

where $D_{train} \neq D_{test}$

Simple train-test procedure



- Provided large enough dataset D drown from p_{data}
- Arrange samples in random order
- Split dataset in two: D_{train} and D_{test}
- Use D_{train} to find the best predictor f
- Use D_{test} to evaluate the generalization performance of f

Source: P. Vincent

Model selection: choosing a function family F

Ex. of parameterized function families for a regression task

- Constant predictor $f_\theta(x) = b$

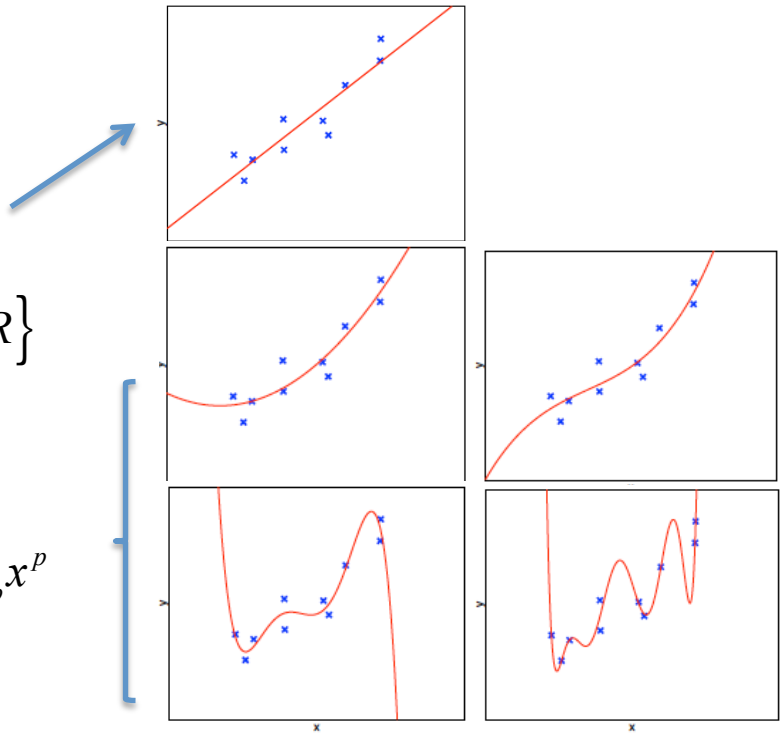
  where $\theta = b$

- Linear (affine) predictor: $f_\theta(x) = w^t x + b$

  where $\theta = \{w \in R^d, b \in R\}$

- Polynomial predictor of degree p

$$f_\theta(x) = b + a_1 x + a_2 x^2 + \ldots + a_p x^p$$



Capacity of a learning algorithm

- How 'big, rich, flexible expressive, complex' the function family is defines what is informally called the 'capacity' of the ML algorithm
  - e.g. capacity ($F_{polynomial3}$) > capacity (F_{linear})
- There are several formal measures of capacity (e.g. VC-dimension)
- One estimate is the number of adaptable parameters (with exception of linear mappings)
- The effective capacity of a ML algorithm is controlled by
  - choice of the algorithm (**family** F)
  - **hyper-parameters** that specify F
  - hyper-parameters of **regularization** schemes
  - hyper-parameters that control **early-stopping**

Source: P. Vincent

Popular classifiers: their parameters and hyperparameters

Algorithm	Capacity-control hyperparameters	Learned parameters
Logistic regression (L2 regularized)	Strength of L2 regularizer	w, b
Linear SVM	C	w,b
Kernel SVM	C, kernel choice & params (s for RBF, degree for polynomial)	Support vector weights a
Neural network	Layer sizes, early stop, …	Layer weight matrices
Decision tree	Depth	The tree (with index and threshold of variables)
K-nearest neighbors	K, choice of metric	Memorizes trainset

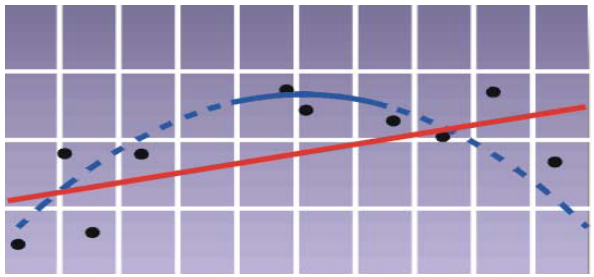
Source: P. Vincent

Tuning capacity

Capacity must be optimally tuned to ensure good generalization

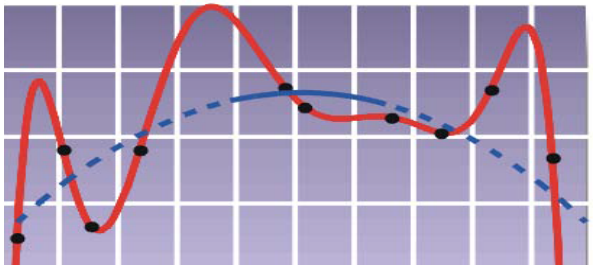
- by choosing algorithm and hyperparameters
- to avoid under-fitting and over-fitting

Example: 1D regression with a polynomial predictor



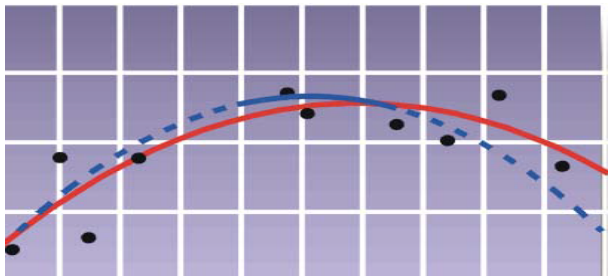
capacity too low

⇨ under-fitting



capacity too high

⇨ over-fitting



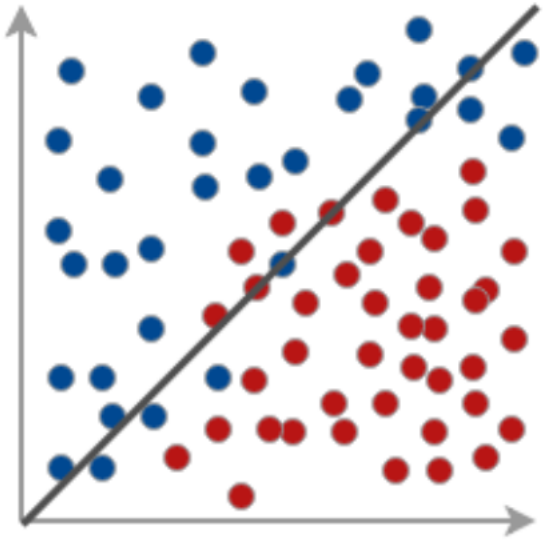
optimal capacity

⇨ good generalisation

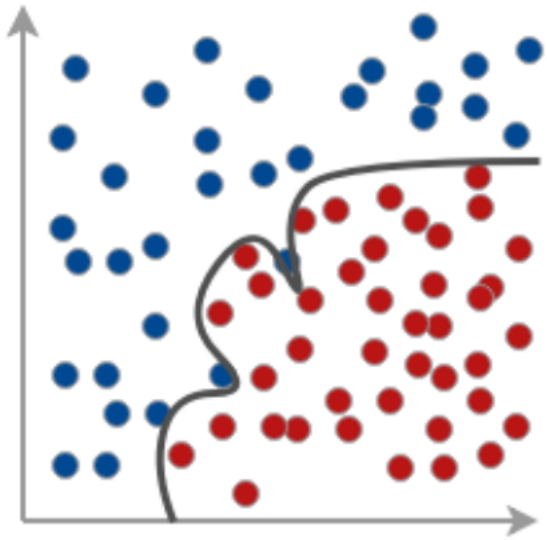
performance on training set is not a good estimate of generalization

Source: P. Vincent

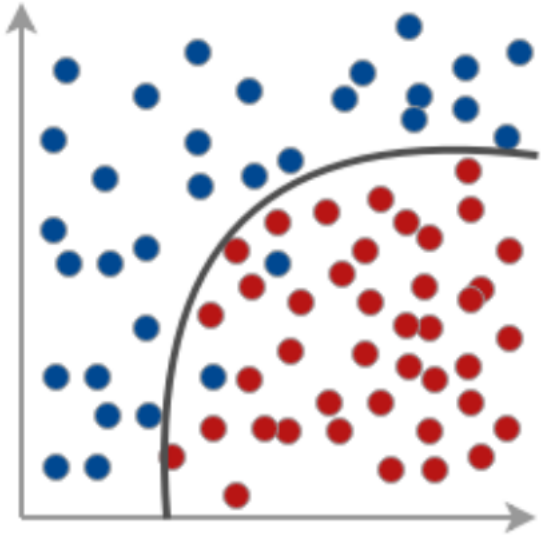
Ex: 2D classification



Function family too poor
Capacity too low for this problem
⇨ **Under-fitting**



Function family too rich
Capacity too high for this problem
⇨ **Over-fitting**

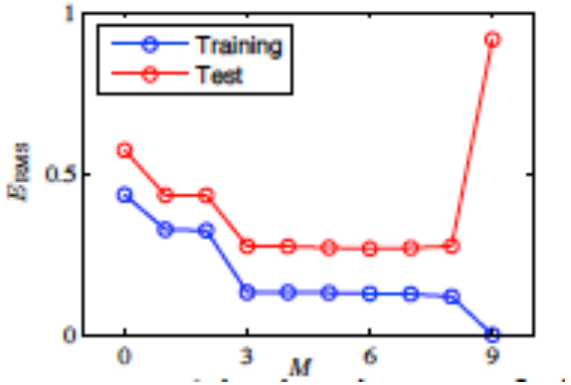


Optimal capacity for this problem
Best generalization
(on future test points)

Overfitting

- A general, very important problem for all machine learning algorithms
- We can find a function that predicts perfectly the training data but does not generalize well to new data

Typical overfitting plot (regression problem)

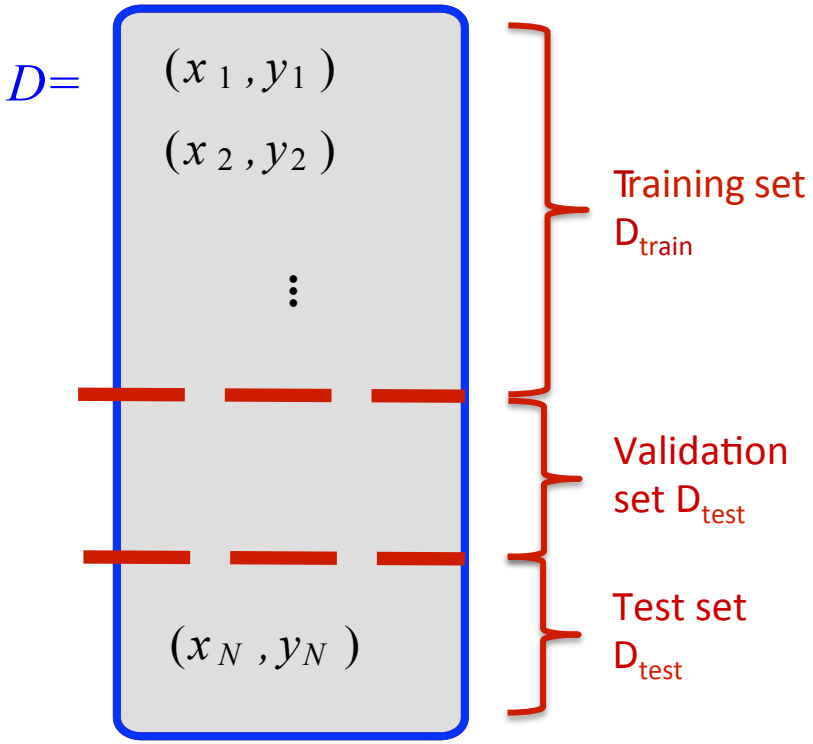


- The training error decreases with the degree of the polynomial M (capacity of the model)
- The test error (measured on independent data) decreases at first and then starts increasing
- **Cross validation helps us**
  - Find a good hypothesis class (M in our case), using a validation set of data
  - Report unbiased results, using a test set, untouched during either training or validation

Cross-validation

Make sure examples are in random order

Split data D in 3: D_{train} D_{valid} D_{test}



- A general procedure for estimating the true error of a predictor
- Data is split into subsets
  - Training and Validation set used only to find the right predictor (optimize hyperparameters)
  - A Test set used to report the prediction error of the algorithm
- The sets must be disjoint
- The process is repeated several times, and the results are averaged to provide error estimates

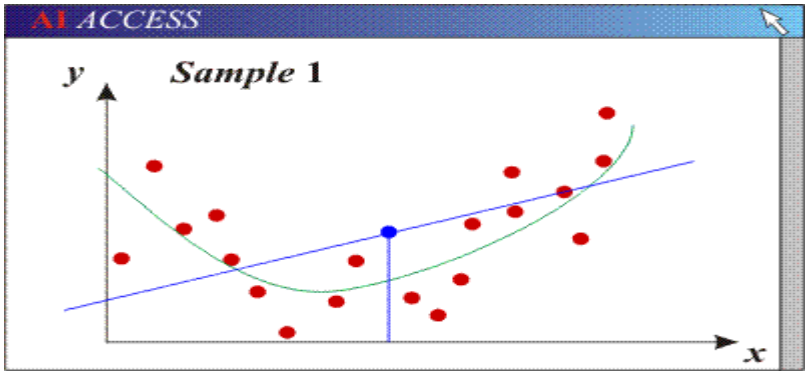
Achieving good generalization

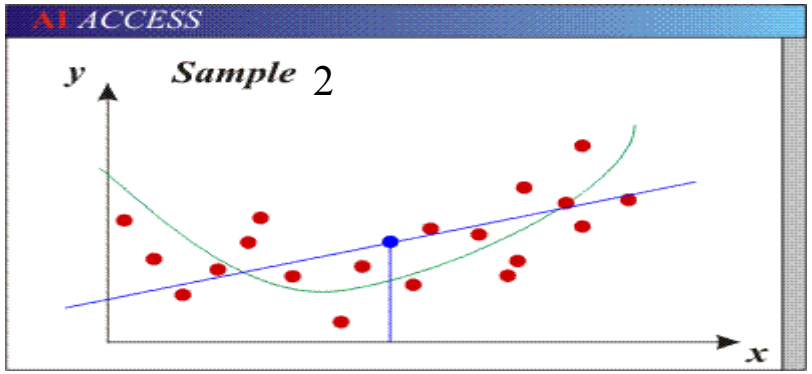
- Consideration 1: Bias
  - How much does the average model over all training sets differ from the true model? (or how well does your model fit the observed data? accuracy of the model)
  - It may be a good idea to accept some fitting error, because it may be due to noise or other "accidental" characteristics of one particular training set

- Consideration 2: Variance
  - How much does the model estimated from different training sets differ from each other? Or how robust is the model to the selection of a particular training set?

Bias/variance tradeoff

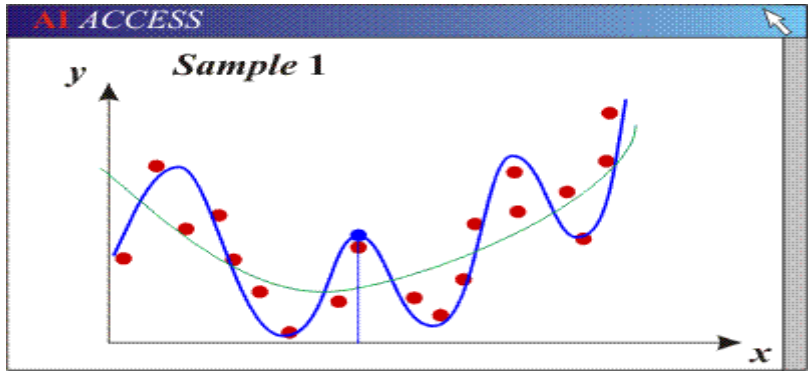
- Models with too few parameters may not fit the data well (high bias) but are consistent across different training sets (low variance)
- Generalization error is due to *underfitting:* model is too "simple" to represent all relevant class characteristics

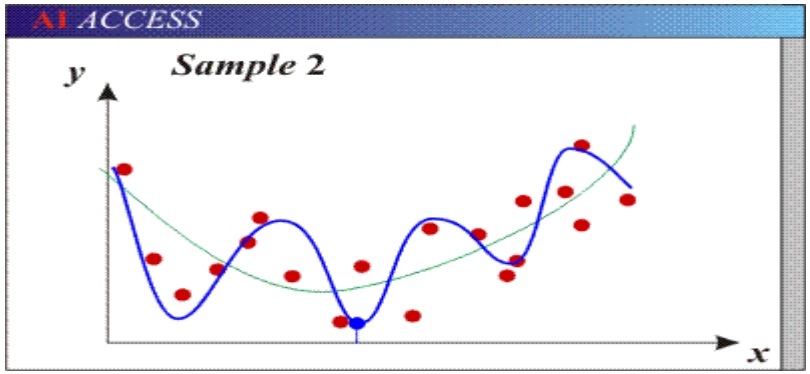




Bias/variance tradeoff

- Models with too many parameters may fit the training data well (low bias), but are sensitive to choice of training set (high variance)
- Generalization error is due to *overfitting:* model is too "complex" and fits irrelevant (noise) characteristics in the data

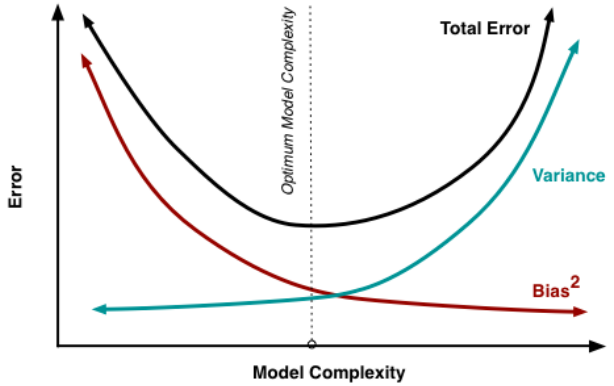




Underfitting and overfitting

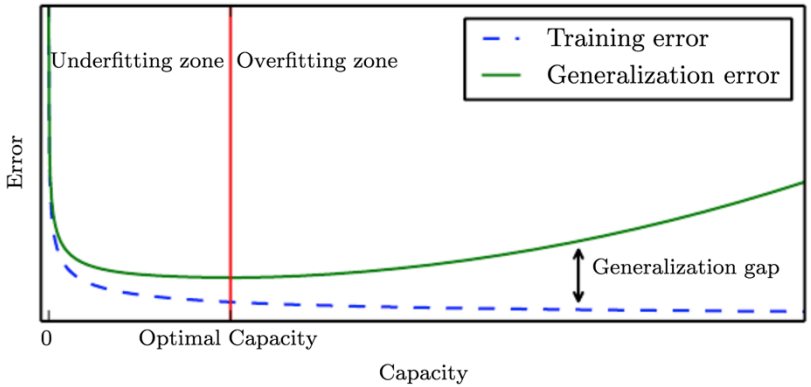
Underfitting: high bias and low variance

- How to recognize underfitting?
  - High training error and high test error
- How to deal with underfitting?
  - Find a more complex model



Overfitting: low bias and high variance

- How to recognize overfitting?
  - Low training error, but high test error
- How to deal with overfitting?
  - Get more training data
  - Decrease the number of parameters in your model
  - Regularization: penalize certain parts of the parameter space or introduce additional constraints to deal with a potentially ill-posed problem



Summary

- Learning approaches: **supervised**, **unsupervised**, **reinforcement learning**
- Machine learning algorithms make choices of function family, error function and optimization procedure
- Capacity, **overfitting**, **underfitting**
- All algorithms are affected by bias-variance trade-off (too much variance = overfitting)
- **Crossvalidation** as a general procedure for estimating the true error of a predictor (disjoint training-validation & test sets)

Bibliograpy

- D. Precup, Introduction to machine learning. Deep Learning Summer School, Montreal, 2017
- P. Vincent, Introduction to machine learning. Deep Learning Summer School, Montreal, 2015
- C. Bishop, Pattern recognition and machine learning
- R. Duda and P. Hart, Pattern classification