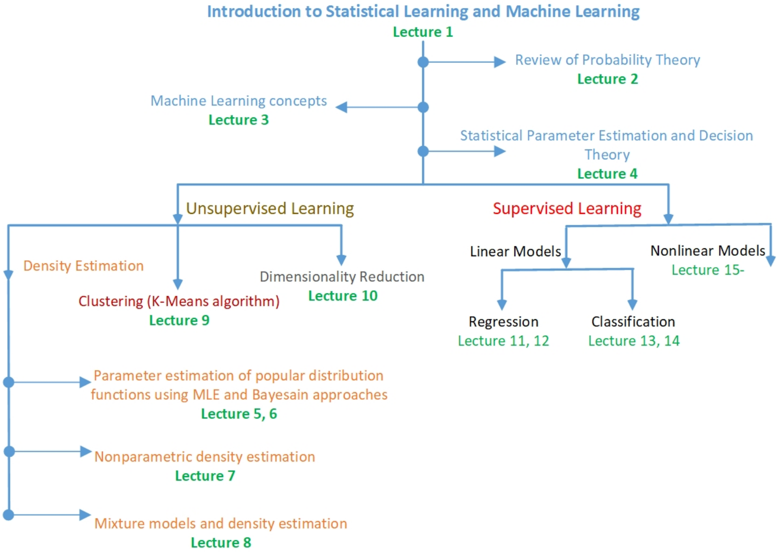
Statistical Learning and Machine Learning

Lecture 3 - Curve Fitting and Model Selection

August, 2025

Course overview and where do we stand

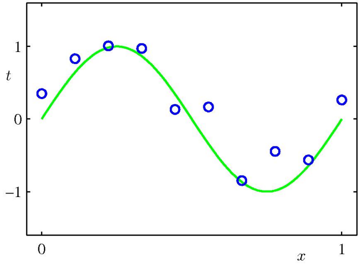


Objectives of the Lecture

- Using the example of polynomial curve fitting (regression) to illustrate key concepts involved in supervised learning process
- The related concepts include:
  1. using training, validation and testing data sets for optimal model parameter estimation
  2. model complexity in learning algorithms and systematic ways to choose optimal models in terms of complexity (e.g., regularization)
  3. curse of (higher) dimensionality in learning algorithms

Example: Polynomial Curve fitting

- Consider that we observe a real-valued variable x and we want to use it to predict the value of a real-valued target value t.
- We use synthetically generated data to know the precise model that generated the data.



Green curve: underlying process $sin(2\pi x)$.
Blue circles: subset of training data consisting of $N = 10$ pairs $\{(x_i, t_i)\}_{i=1}^{10}$, where $t_i = sin(2\pi x_i) + noise$

Example: Polynomial Curve fitting

Suppose we are given:

- a training set of N observations of x, which we denote by $\boldsymbol{x} = [x_1, \dots, x_N]^T$ along with the corresponding target values, $\boldsymbol{t} = [t_1, \dots, t_N]^T$

Our goal is to:

- Use the N data points $\boldsymbol{x}$ and the corresponding target values $\boldsymbol{t}$ to *learn* a function $\tilde{t} = y(\tilde{x})$ which can *predict* the target value for a new point $\tilde{x}$
- Model: The model/function y will be *parametric*, meaning that it uses a number of parameters $\boldsymbol{w}$ which can be *estimated* using the training data $\{\boldsymbol{x}, \boldsymbol{t}\}$

Ideally, our model should correspond to the underlying function (in our example, $sin(2\pi x)$.

Example: Polynomial Curve fitting

- Consider a polynomial parametric model w.r.t. the input variable x i.e.,

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

  where M is the *order* of the polynomial and x^j denotes x raised to the power j. The polynomial coefficients $w_0, \ldots, w_M$ can be collected in a vector form $\boldsymbol{w} \in \mathbb{R}^{(M+1)}$ and $\boldsymbol{x} = [1, x, x^2, \ldots, x^M]$
- The function $y(x, \boldsymbol{w})$ is a linear function of the unknown parameters $\boldsymbol{w}$ - an example of **linear models**.

Example: Polynomial Curve fitting

- The coefficients $\boldsymbol{w}$ fully specify our model!
- Estimating the coefficients/parameters $\boldsymbol{w}$ can be done by minimizing an *error function*, like:

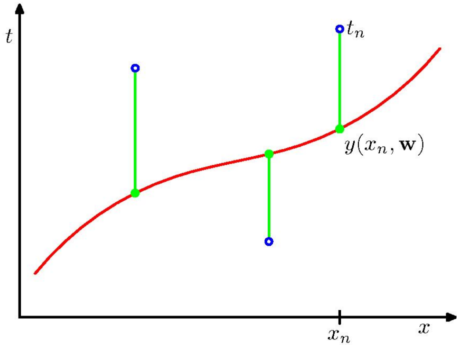
$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\Big(y(x_n, \boldsymbol{w}) - t_n\Big)^2$$

$E(\boldsymbol{w})$ is called *sum-of-squares error* function:

1. it is non-negative: $E(\boldsymbol{w}) \geq 0$
2. it becomes equal to zero for $y(x_n, \boldsymbol{w}) = t_n$, for $n = 1, \ldots, N$

Example: Polynomial Curve fitting

Intuition behind minimizing the sum-of-squares error function



Imagine multiple 'candidate' red curves/functions corresponding to different $\boldsymbol{w}$ and that optimization process chooses one that minimizes $E(\boldsymbol{w})$.

Example: Polynomial Curve fitting

How is the **Optimization** of $E(\boldsymbol{w})$ performed:

- $E(\boldsymbol{w})$ is a quadratic function of $\boldsymbol{w}$ (it has one, so called *global*, minimum)
- The optimal parameter vector $\boldsymbol{w}^*$ is obtained by setting $\frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}} \to 0$
- By setting:
  - $\boldsymbol{x}_n = [1, x_n, x_n^2, \ldots, x_n^M]^T \in \mathbb{R}^{(M+1)\times 1}$
  - $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix} \in \mathbb{R}^{N\times(M+1)}$

  we can write $E(\boldsymbol{w})$ in its *matrix form*:

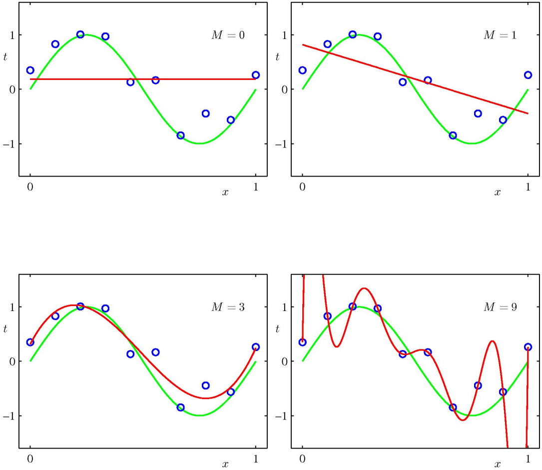
$$\begin{aligned} E(\boldsymbol{w}) &= \|\boldsymbol{t} - \boldsymbol{X}\boldsymbol{w}\|^2 \\ &= \boldsymbol{t}^T\boldsymbol{t} - 2\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{t} + \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w}. \end{aligned}$$

- The optimal value $\boldsymbol{w}^*$ is:

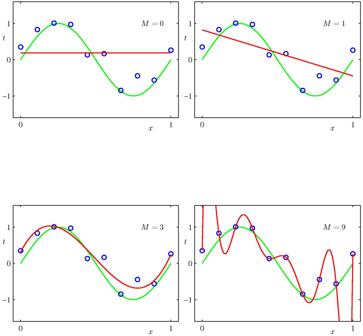
$$\frac{\partial E(\mathbf{w})}{\partial \boldsymbol{w}} \to 0 \quad \Rightarrow \quad \boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{t}$$

Example: Polynomial Curve fitting

Model Selection: Which of the models do you believe is the best in terms of predicting correct output on i) training data; ii) new values of input?



Example: Polynomial Curve fitting



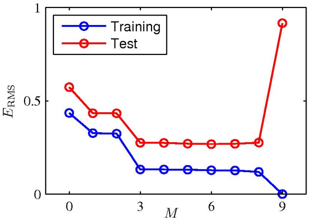
- Models with $M = 0$, $M = 1$ give poor fit for the data
- Model with $M = 9$ give excellent fit to the training data but oscillates wildly in between the blue points (training data) - an example of **over-fitting**.
- Model with $M = 3$ gives good approximation of the underlying function.

Example: Polynomial Curve fitting

Generalization: Making accurate predictions on 'new' data is the main goal. To measure the generalization performance of $y_n(x, \boldsymbol{w})$:

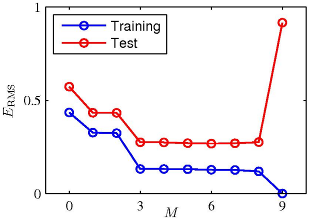
- we generate and use **test** data set
- **test** data set is generated in the same way as training data but is not used in training phase (for calculation of $\boldsymbol{w}$)
- we evaluate the performance of each choice of M on the test data using a convenient measure (root-mean-square error):

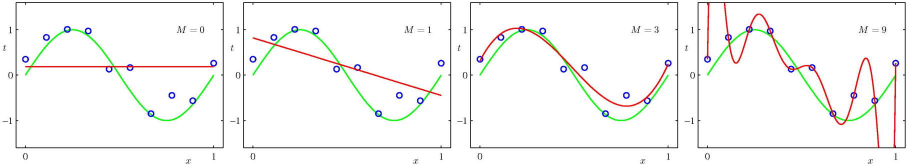
$$E_{RMS} = \sqrt{2E(\mathbf{w}^*)/N}$$



Example: Polynomial Curve fitting

Analyse the graphs and make observations on the behaviour of the error curves for 'training' and 'test' data for: i) $M < 3$; ii) $3 \leq M \leq 8$ iii) $M > 8$





Example: Polynomial Curve fitting

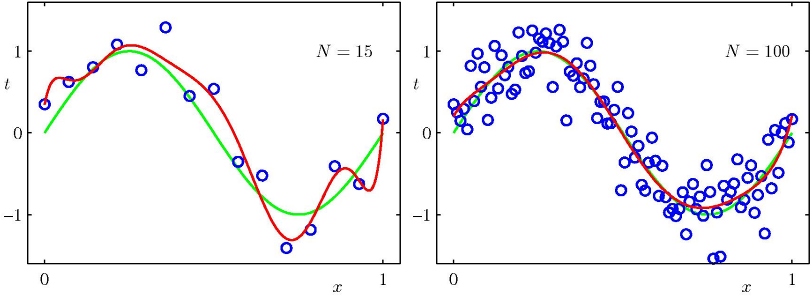
Insight into **Over-fitting** phenomenon for large values of M.

- For $M = 9$, the values of calculated parameters $\mathbf{w}$ are very large
- Those large values lead to massive oscillations that are undesirable.

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Example: Polynomial Curve fitting

Possible remedy of Over-fitting: Use larger training data set!



This led to the advocation of a rough heuristic that number of training data must be at least 5 to $10\times$ more than the number of parameters in the model. What's wrong with that?

Example: Polynomial Curve fitting

A better remedy to over-fitting: Regularization

- **Main Idea**: 'Forcing' the model parameters, $\boldsymbol{w}$, NOT to take large values via constrained optimization
- **The new cost function**: summation of training error and the norm of model parameter vector $\boldsymbol{w}$

$$\tilde{E}(\boldsymbol{w}) = \frac{1}{2}\sum_{n+1}^{N}\Big(y(\boldsymbol{x}_n, \boldsymbol{w}) - t_n\Big)^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

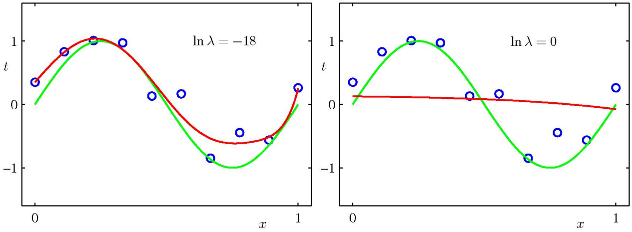
where

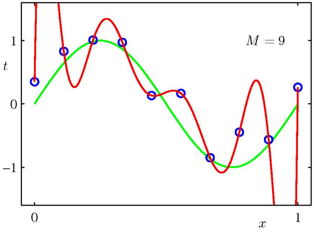
$$\|\boldsymbol{w}\|^2 = w_0^2 + w_1^2 + \dots w_M^2$$

- Solution:

$$w_{reg}^{*} = (\lambda \boldsymbol{I} + \boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{t}$$

Results of regularization of model parameters for different values of λ

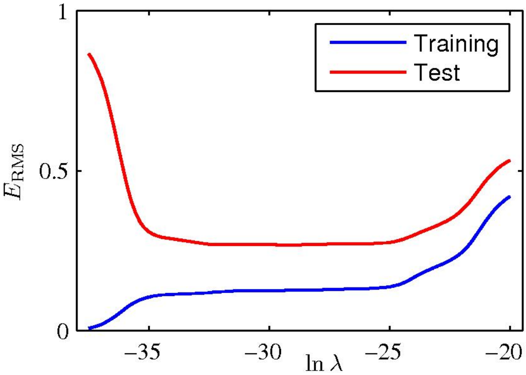




	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Example: Polynomial Curve fitting

RMSE curves for training and testing data sets for different values of the regularization parameter λ



Increasing values of λ correspond to lower complexity of models.

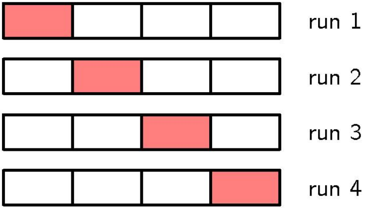
Model Selection in Learning Algorithms

- In the curve fitting example, the performance of our learning algorithm/model depended critically on the model complexity (poor fit, over-fitting).
- How to systematically estimate parameters defining model complexity (order of the polynomial M or regularization parameter *lambda* in the curve fitting example) in order to achieve the best performance.
- **Validation Set**: independent data set used just for tuning/optimizing hyper-parameters
  1. Train multiple models (each for a different combination values for the hyper-parameters of the model) using the training set
  2. Measure the performance of the multiple models on the validation set
  3. Select the model providing the best performance on the validation set
  4. Test the best model on the 'test' data

Model Selection in Learning Algorithms

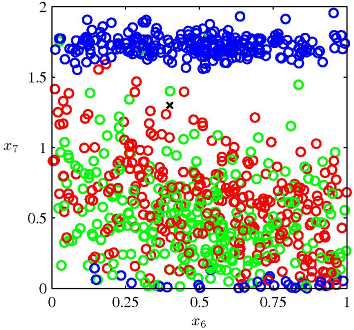
Cross-validation is used when the supply of data for training and testing is limited:

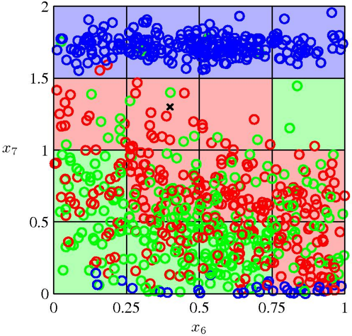
1. We split the training set in S (mutually exclusive) subsets (S-fold cross validation).
2. For each model (corresponding to a combination of hyper-parameter values):
    1. we train the model S times using $S-1$ sets
    2. we test it with the remaining set
    3. we calculate the average performance over all S experiments and the corresponding standard deviation
3. We select the model providing the best overall performance



The Curse of Dimensionality

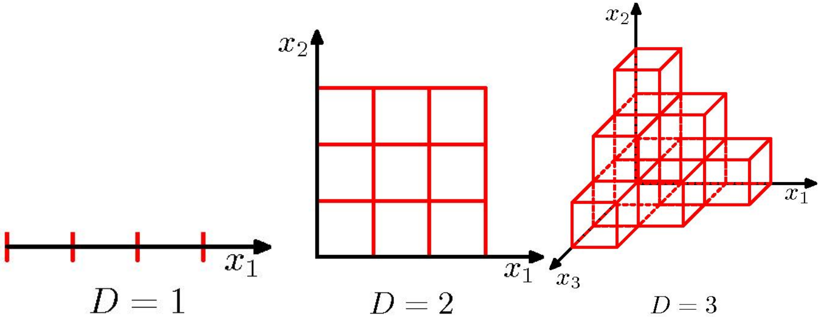
- The use of high-dimensional data in machine learning models poses some serious challenges.
- Example: problem of classification of input data into 3 classes (red, blue, green), based on training data shown below.





The Curse of Dimensionality

- Extending the idea above to D-dimensional spaces with $D \gg 1$ leads to an exponential growth of the *hyper-cubes* needed to cover the space.
- We would need an exponentially large quantity of training data in order to ensure that the cells are not empty!



The Curse of Dimensionality

- Consider the polynomial curve fitting example, the input x was single dimensional.
- Assume that input was D-dimensional i.e., $\boldsymbol{x} = [x_1, x_2, \ldots, x_D]$, then for a polynomial model with $M = 3$, we get

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + \sum_{i=1}^{D} w_i x_i + \sum_{i=1}^{D}\sum_{j=1}^{D} w_{ij} x_i x_j + \sum_{i=1}^{D}\sum_{j=1}^{D}\sum_{k=1}^{D} w_{ijk} x_i x_j x_k$$

The number of model coefficients in this case are D^3. For a model of order M, the number of model coefficients would grow to D^M. Hence, the number of model parameters can become prohibitively large even for moderate values of D and M.