FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
NATURWISSENSCHAFTLICHE
FAKULTÄT

Introduction to Supervised Learning

Lecture "Mathematics of Learning" 2022

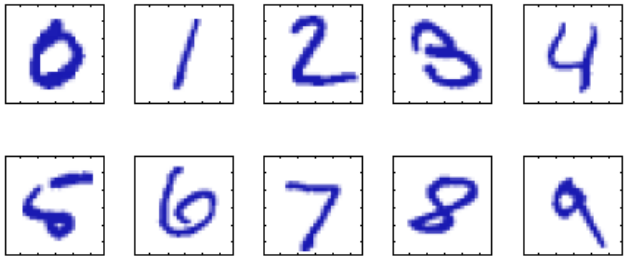
Andreas Bärmann
Friedrich-Alexander-Universität Erlangen-Nürnberg

WIRTSCHAFTS
MATHEMATIK

Introduction Supervised Learning

(see Bishop Pattern Recognition book, chapter 1)
...back to our lecture topics: now supervised learning:

Figure 1.1 Examples of hand-written digits taken from US zip codes.



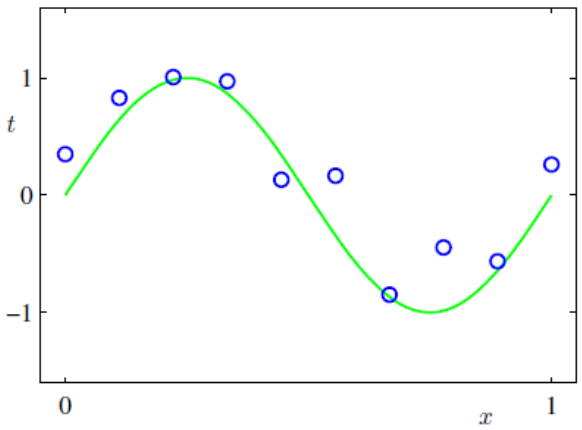
- large *training* data set with N points $\{x_1, \ldots, x_N\}$
- labels / categories (e.g. digits in handwriting, different objects...) are known in advance ('labeled data'), stored in vector t ('target') for each data point
- use training data for tuning parameters of an adaptive model
- *training phase or learning phase* results in function $y(x)$ that takes data point x as input, returns $y(x)$ that corresponds to a specific target / label
- *generalization*: categorizing additional data contained in some *test set* is the main goal

Introduction Supervised Learning

- *feature extraction*: reduce difficulty of the problem by reduction of data through preprocessing (scaling,...) and/or dimensionality reduction
- digit recognition is *classification problem*

Regression: Polynomial Curve Fitting

Figure 1.2 Plot of a training data set of $N = 10$ points, shown as blue circles, each comprising an observation of the input variable x along with the corresponding target variable t. The green curve shows the function $\sin(2\pi x)$ used to generate the data. Our goal is to predict the value of t for some new value of x, without knowledge of the green curve.



- use data x to predict value of real target value $\hat{y}$
- In this example, x is only one dimensional. Later, we will study the problem in higher dimensions.
- Assume there is some underlying regularity, i.e. there is something to 'learn'.
- given: training set $\mathbf{x} = (x_1, \ldots, x_N)$, $\hat{\mathbf{y}} = (\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_N)$

Regression: Polynomial Curve Fitting

- determine a fit with a polynomial function of form

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_D x^D = \sum_{j=0}^{D} w_j x^j,$$

  where D is the order of the polynomial, real coefficients w_j.
- monomials $x, x^2, x^3, \ldots$, are also called *basis functions*.
- Although $y(x, \mathbf{w})$ itself is a polynomial, it is only *linear* in the unknown coefficients $\mathbf{w}$ that need to be determined by fitting the polynomial to our training data
- We determine $\mathbf{w}$ by minimizing a quadratic *loss function* that measures the error between data and prediction, e.g.:

$$L(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, \mathbf{w}) - \hat{y}_n)^2$$

- This is also called *method of least squares*.

Regression: Polynomial Curve Fitting

- Determine optimum value of the coefficients $\mathbf{w}$ such that loss function $L(\mathbf{w})$ is minimized.
- In order to do this, we determine critical points where the gradient $\nabla_{\mathbf{w}} L(\mathbf{w})$ w.r.t. $\mathbf{w}$ is zero.
- This gradient is linear in $\mathbf{w}$, and solving $\nabla_{\mathbf{w}} L(\mathbf{w}) = 0$ yields solution $\mathbf{w}^\star$. This corresponds to the fitted polynomial $y(x, \mathbf{w}^\star)$.
- We will later discuss assumptions for uniqueness of the coefficients. For now, we assume $\mathbf{w}^\star$ is unique, so that also the fitted polynomial is uniquely defined.
- We note that the choice of D is important, and many methods exist to decide on good values.

Potential Problem of Overfitting!

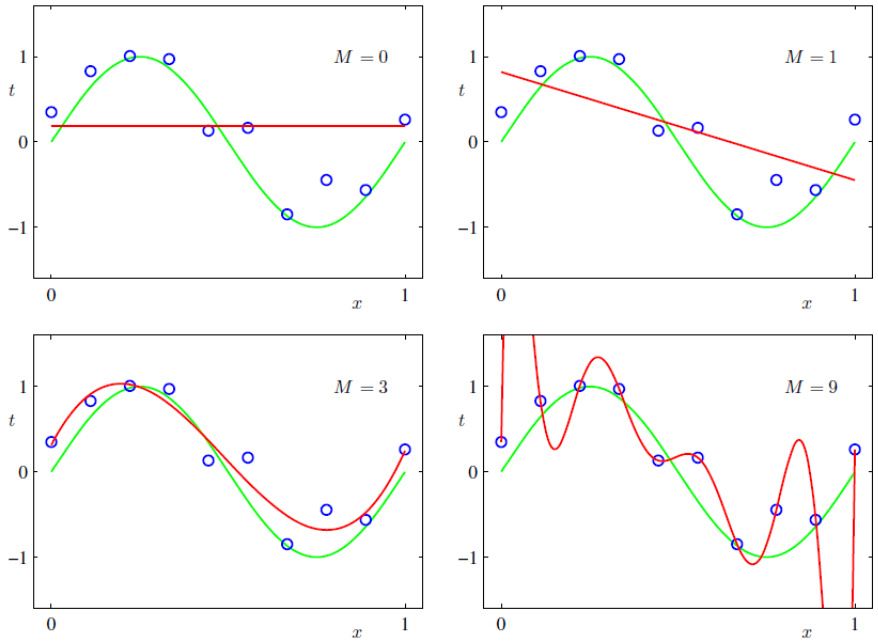
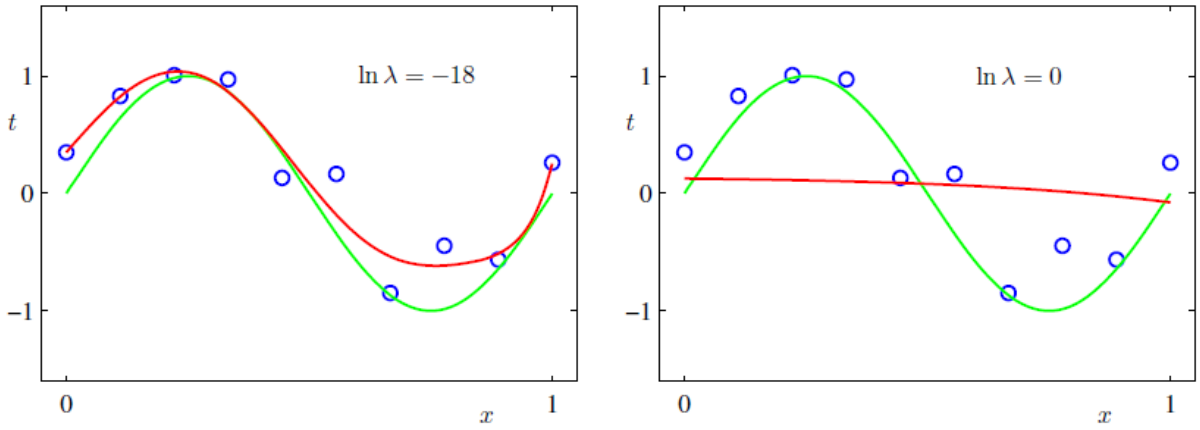


Figure 1.4 Plots of polynomials having various orders M, shown as red curves, fitted to the data set shown in Figure 1.2.

$D = 3$ is a good choice. While a very large value of D yields zero loss, it leads to *overfitting*, i.e. to a poor representation of the underlying structure (here a cosine function) that is distracted by noise and measurement errors. Thus, fitted function will generalize poorly to new and unseen data.
When learning from data, overfitting has to be avoided!

Avoid Overfitting by Regularization

- An underlying problem with too large D consists in the fact that there are typically very large (both positive and negative) coefficients in the fitted polynomial.
- As a way out: We add a penalty term in order to avoid large coefficients.
  $\tilde{L}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(y(x_n, \mathbf{w}) - t_n)^2 + \frac{\lambda}{2}||\mathbf{w}||^2$
- the value of λ governs the importance of regularization
- This loss function can still be minimized in closed form.



igure 1.7 Plots of $M = 9$ polynomials fitted to the data set shown in Figure 1.2 using the regularized error unction (1.4) for two values of the regularization parameter λ corresponding to $\ln\lambda = -18$ and $\ln\lambda = 0$. The ase of no regularizer, i.e., $\lambda = 0$, corresponding to $\ln\lambda = -\infty$, is shown at the bottom right of Figure 1.4.

Potential Issue: Curse of Dimensionality

Suppose the input has M many input variables $(x_{j1}, \ldots, x_{jM})$, then a general polynomial of order D will have a number of coefficients in $O(M^D)$, which grows very quickly.
Thus, computational efficiency has to be taken into account.

Example for a Polynomial Fit

A polynomial fit requires alphanumeric target values. However, if there are only two classes, also classification problems on categorical variables can be solved. Training data on inputs (X_1, X_2): target t takes either value $Y = 0$ for value blue or $Y = 1$ for value orange. Targets denoted by $\hat{Y}$.

$$t = \begin{cases} \text{orange} & \text{, if } Y > 0.5 \\ \text{blue} & \text{, if } Y \leq 0.5 \end{cases}$$

FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (BLUE = 0, ORANGE = 1), and then fit by linear regression. The line is the decision boundary defined by $x^T\hat{\beta} = 0.5$. The orange shaded region denotes that part of input space classified as ORANGE, while the blue region is classified as BLUE.*

Example for a Polynomial Fit

- set of points in $\mathbb{R}^2$ with $\{x \mid x^\top \hat{\beta} > 0.5\}$ are classified as orange
- $\{x \mid x^\top \hat{\beta} = 0.5\}$ is (here linear) decision boundary

We see several misclassifications on each side.

Can this be avoided?
Assume data was generated by one of the two scenarios:

- scenario 1: in each class, training data was generated from Gaussian distribution, uncorrelated components, different means
- scenario 2: data comes from mixture of 10 low-variance Gaussians, means themselves distributed as Gaussian.

in case of scenario 1: linear decision boundary is best we can do, estimate is almost optimal (see later...), region of overlap is inevitable.
in case of scenario 2: linear boundary is unlikely to be optimal, should be non-linear & disjoint and more difficult.
The so-called *nearest-neighbour method* is better suited for scenario 2, see next.

Classification via Nearest-Neighbour Methods

We use a similar idea as in unsupervised learning for k-means clustering. Here: For estimating the targets, we use observations in training set that are 'closest' in input space, measured in some metric, e.g. in Euclidean distance. Let $N_k(x)$ neighbourhood of x defined by k closest points x_i in training set. k-nearest neighbour fit to estimate target $\hat{Y}$:

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

$$t = \begin{cases} \text{orange} & , \text{if } \hat{Y} > 0.5 \\ \text{blue} & , \text{if } \hat{Y} \leq 0.5 \end{cases}$$

in general: We often use majority voting for classification problems, i.e. the group is assigned to the target value that the majority of the data points in the neighbourhood have.

Nearest-Neighbour Methods

15-nearest-neighbour averaging:

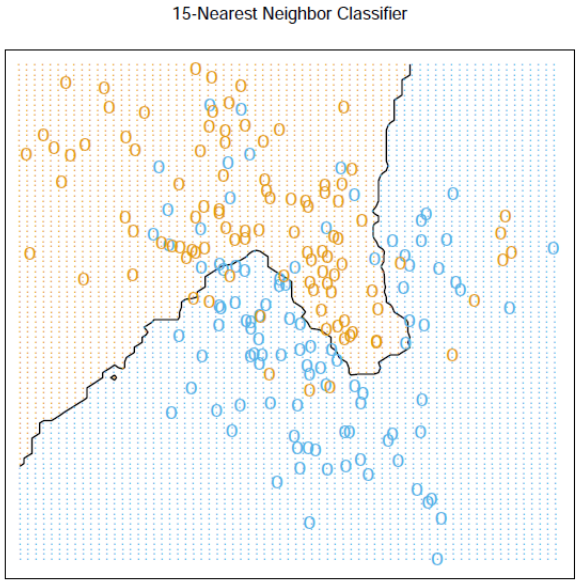


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

irregular boundary

Nearest-Neighbour Methods

however: for $k = 1$, the error (i.e. sum of Euclidean distances) will always be zero, see figure.

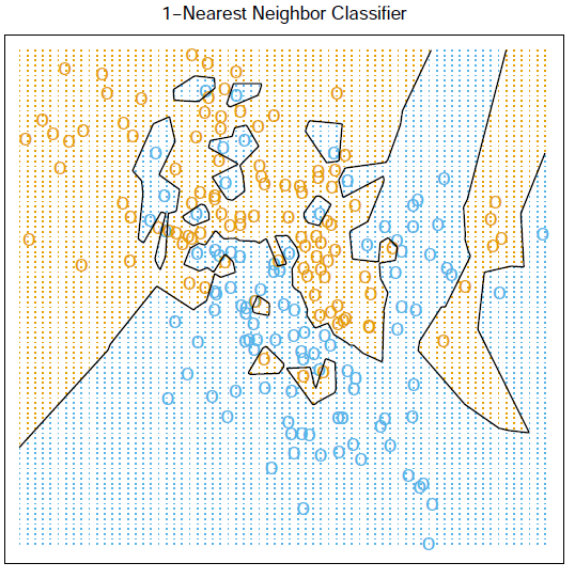


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$), *and then predicted by* 1*-nearest-neighbor classification.*

the error will increase with value of k.

Comparison of Least Squares with Nearest Neighbours

least squares: linear decision boundary is smooth and stable, relies on assumption that a linear function for the boundary is appropriate. *This is an a priori assumption on the model that may or may not be justified in practice.*
nearest neighbours: does not assume anything on underlying data, can adapt. however, is unstable
each method has its situations in which it works best. least squares for data resemble scenario 1, nearest neighbour scenario 2.
This ends the first introduction into supervised learning. We will now look into different settings and algorithms in more detail.