Introduction to Machine Learning

Milan Straka

October 07, 2019

Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Machine Learning

A possible definition of learning from Mitchell (1997):

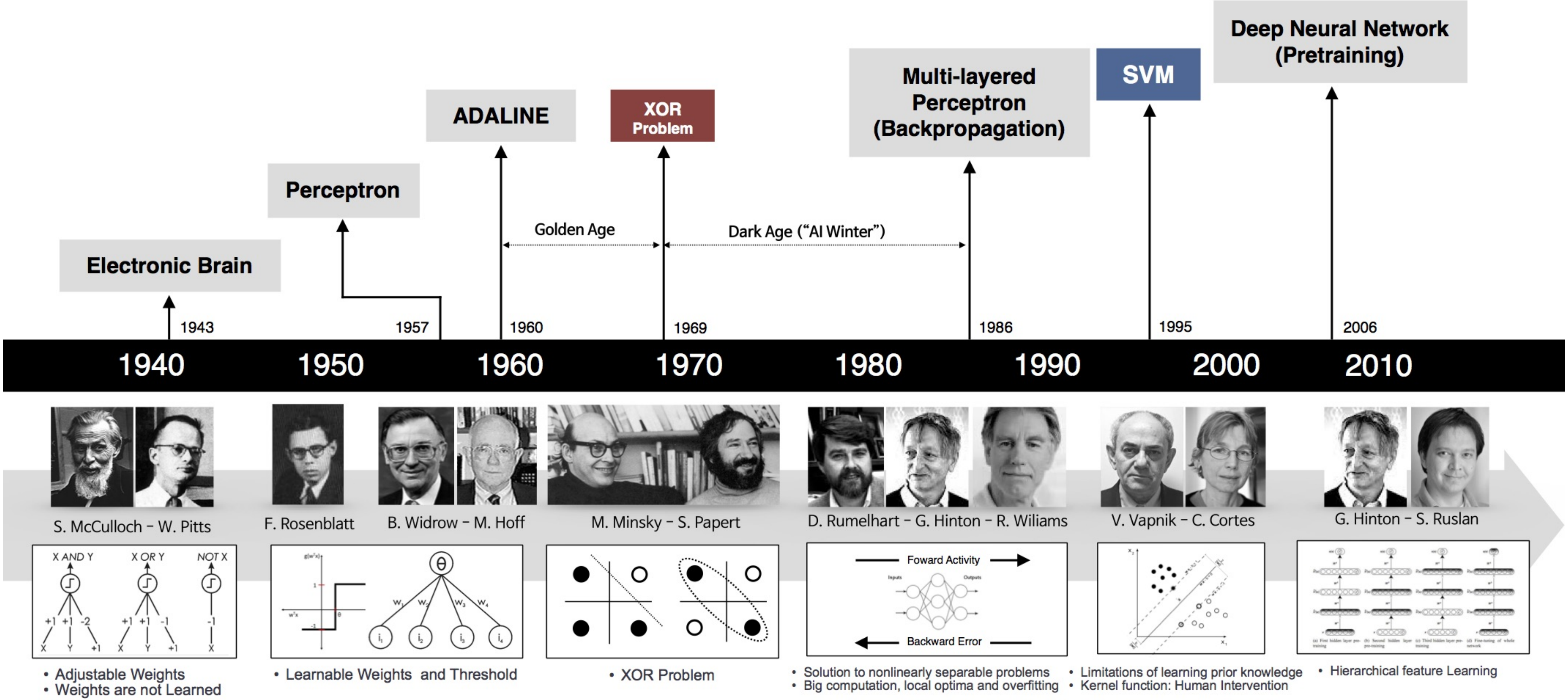
> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

- Task T
  - *classification*: assigning one of k categories to a given input
  - *regression*: producing a number $x \in \mathbb{R}$ for a given input
  - *structured prediction*, *denoising*, *density estimation*, …
- Experience E
  - *supervised*: usually a dataset with desired outcomes (*labels* or *targets*)
  - *unsupervised*: usually data without any annotation (raw text, raw images, …)
  - *reinforcement learning*, *semi-supervised learning*, …
- Measure P
  - *accuracy*, *error rate*, *F-score*, …

Deep Learning Highlights

- Image recognition
- Object detection
- Image segmentation
- Human pose estimation
- Image labeling
- Visual question answering
- Speech recognition and generation
- Lip reading
- Machine translation
- Machine translation without parallel data
- Chess, Go and Shogi
- Multiplayer Capture the flag

Introduction to Machine Learning History



- Adjustable Weights
- Weights are not Learned
- Learnable Weights and Threshold
- XOR Problem
- Solution to nonlinearly separable problems
- Big computation, local optima and overfitting
- Limitations of learning prior knowledge
- Kernel function: Human Intervention
- Hierarchical feature Learning

https://www.slideshare.net/deview/251-implementing-deep-learning-using-cu-dnn/4

Machine and Representation Learning

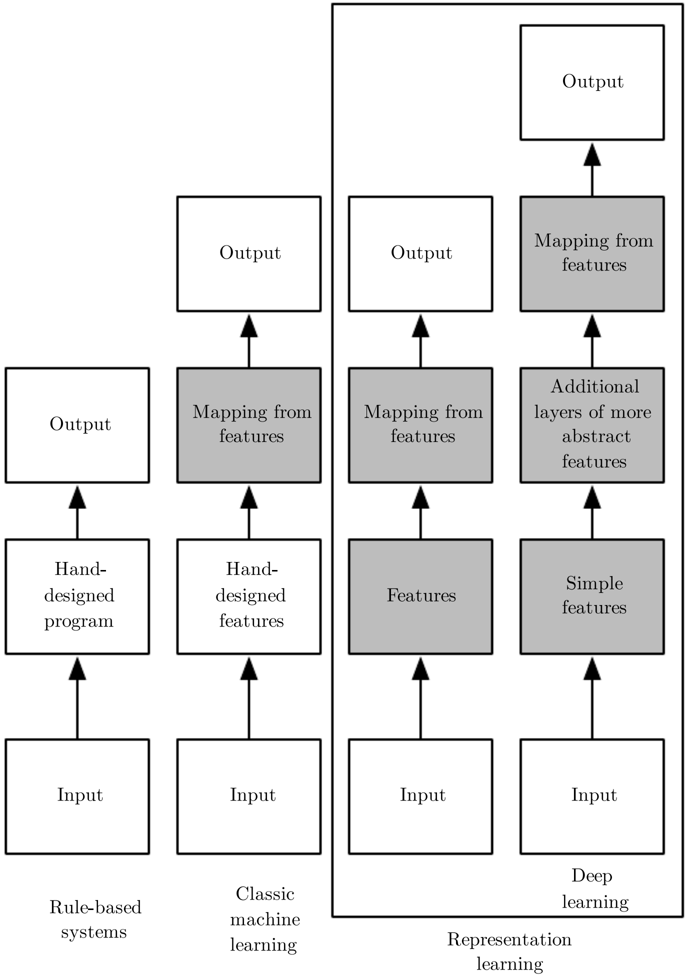


Figure 1.5, page 10 of Deep Learning Book, http://deeplearningbook.org.

Basic Machine Learning Settings

Assume we have an input of $\boldsymbol{x} \in \mathbb{R}^d$.

Then the two basic ML tasks are:

1. **regression**: The goal of a regression is to predict real-valued target variable $t \in \mathbb{R}$ of the given input.
2. **classification**: Assuming we have a fixed set of K labels, the goal of a classification is to choose a corresponding label/class for a given input.
   - We can predict the class only.
   - We can predict the whole distribution of all classes probabilities.

We usually have a **training set**, which is assumed to consist of examples of $(\boldsymbol{x}, t)$ generated independently from a **data generating distribution**.

The goal of *optimization* is to match the training set as well as possible.

However, the goal of *machine learning* is to perform well on *previously unseen* data, to achieve lowest **generalization error** or **test error**. We typically estimate it using a **test set** of examples independent of the training set, but generated by the same data generating distribution.

Notation

- a, $\boldsymbol{a}$, $\boldsymbol{A}$, A: scalar (integer or real), vector, matrix, tensor
- a, $\mathbf{a}$, $\mathbf{A}$: scalar, vector, matrix random variable
- $\frac{df}{dx}$: derivative of f with respect to x
- $\frac{\partial f}{\partial x}$: partial derivative of f with respect to x
- $\nabla_{\boldsymbol{x}} f$: gradient of f with respect to $\boldsymbol{x}$, i.e., $\left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \frac{\partial f(\boldsymbol{x})}{\partial x_2}, \ldots, \frac{\partial f(\boldsymbol{x})}{\partial x_n}\right)$

Example Dataset

Assume we have the following data, generated from an underlying curve by adding a small amount of Gaussian noise.

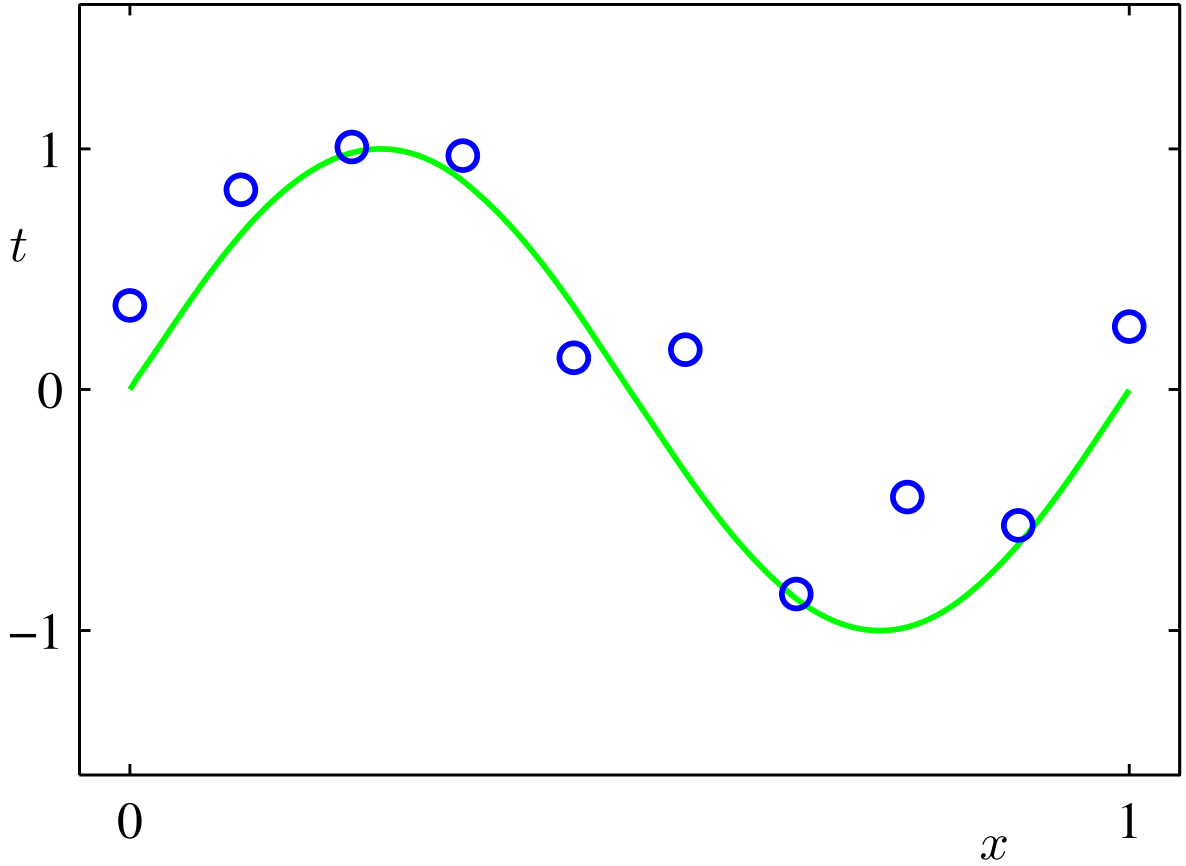


Figure 1.2 of Pattern Recognition and Machine Learning.

Linear Regression

Given an input value $\boldsymbol{x} \in \mathbb{R}^d$, one of the simplest models to predict a target real value is **linear regression**:

$$f(\boldsymbol{x}; \boldsymbol{w}, b) = x_1 w_1 + x_2 w_2 + \ldots + x_D w_D + b = \sum_{i=1}^{d} x_i w_i + b = \boldsymbol{x}^T \boldsymbol{w} + b.$$

The $\boldsymbol{w}$ are usually called *weights* and b is called *bias*.

Sometimes it is convenient not to deal with the bias separately. Instead, we might enlarge the input vector $\boldsymbol{x}$ by padding a value 1, and consider only $\boldsymbol{x}^T \boldsymbol{w}$, where the role of a bias is accomplished by the last weight. Therefore, when we say “weights”, we usually mean both weights and biases.

Separate Bias vs. Padding $\boldsymbol{X}$ with Ones

Using an explicit bias term in the form of $f(x) = \boldsymbol{w}^T\boldsymbol{x} + b$.

$$\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ & \vdots & \\ x_{n1} & x_{n2} \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + b = \begin{bmatrix} w_1x_{11} + w_2x_{12} + b \\ w_1x_{21} + w_2x_{22} + b \\ \vdots \\ w_1x_{n1} + w_2x_{n2} + b \end{bmatrix}$$

With extra 1 padding in $\boldsymbol{X}$ and an additional b weight representing the bias.

$$\begin{bmatrix} x_{11} & x_{12} & 1 \\ x_{21} & x_{22} & 1 \\ & \vdots & \\ x_{n1} & x_{n2} & 1 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} = \begin{bmatrix} w_1x_{11} + w_2x_{12} + b \\ w_1x_{21} + w_2x_{22} + b \\ \vdots \\ w_1x_{n1} + w_2x_{n2} + b \end{bmatrix}$$

Linear Regression

Assume we have a dataset of N input values $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ and targets $t_1, \ldots, t_N$.

To find the values of weights, we usually minimize an **error function** between the real target values and their predictions.

A popular and simple error function is *mean squared error*:

$$\mathrm{MSE}(\boldsymbol{w}) = \frac{1}{N} \sum_{i=1}^{N} \big(f(\boldsymbol{x}_i; \boldsymbol{w}) - t_i\big)^2.$$

Often, *sum of squares*

$$\frac{1}{2} \sum_{i=1}^{N} \big(f(\boldsymbol{x}_i; \boldsymbol{w}) - t_i\big)^2$$

is used instead, because the math comes out nicer.

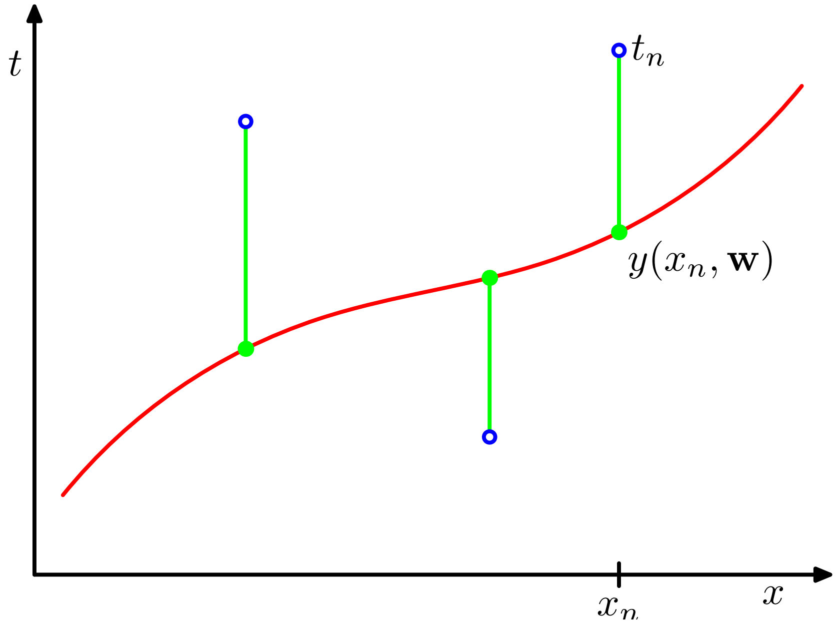


Figure 1.3 of Pattern Recognition and Machine Learning.

Linear Regression

There are several ways how to minimize the error function, but in the case of linear regression and sum of squares error, there exists an explicit solution.

Our goal is to minimize the following quantity:

$$\frac{1}{2}\sum_i^N (\boldsymbol{x}_i^T \boldsymbol{w} - t_i)^2.$$

Note that if we denote $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ the matrix of input values with $\boldsymbol{x}_i$ on a row $\boldsymbol{i}$ and $\boldsymbol{t} \in \mathbb{R}^N$ the vector of target values, we can rewrite the minimized quantity as

$$\frac{1}{2}||\boldsymbol{X}\boldsymbol{w} - \boldsymbol{t}||^2.$$

Linear Regression

In order to find a minimum of $\frac{1}{2}\sum_i^N(\boldsymbol{x}_i^T\boldsymbol{w} - t_i)^2$, we can inspect values where the derivative of the error function is zero, with respect to all weights w_j.

$$\frac{\partial}{\partial w_j}\frac{1}{2}\sum_i^N(\boldsymbol{x}_i^T\boldsymbol{w} - t_i)^2 = \frac{1}{2}\sum_i^N\left(2(\boldsymbol{x}_i^T\boldsymbol{w} - t_i)x_{ij}\right) = \sum_i^N x_{ij}(\boldsymbol{x}_i^T\boldsymbol{w} - t_i)$$

Therefore, we want for all j that $\sum_i^N x_{ij}(\boldsymbol{x}_i^T\boldsymbol{w} - t_i) = 0$. We can write all the equations together using matrix notation as $\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{t}) = 0$ and rewrite to

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{t}.$$

The matrix $\boldsymbol{X}^T\boldsymbol{X}$ is of size $D \times D$. If it is regular, we can compute its inverse and therefore

$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{t}.$$

Linear Regression

Input: Dataset ($\boldsymbol{X} \in \mathbb{R}^{N\times D}$, $\boldsymbol{t} \in \mathbb{R}^N$).
Output: Weights $\boldsymbol{w} \in \mathbb{R}^D$ minimizing MSE of linear regression.

- $\boldsymbol{w} \leftarrow (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{t}$.

The algorithm has complexity $\mathcal{O}(ND^2)$, assuming $N \geq D$.

When the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is singular, we can solve $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{t}$ using SVD, which will be demonstrated on the next lecture.

Linear Regression Example

Assume our input vectors comprise of $\boldsymbol{x} = (x^0, x^1, \ldots, x^M)$, for $M \geq 0$.

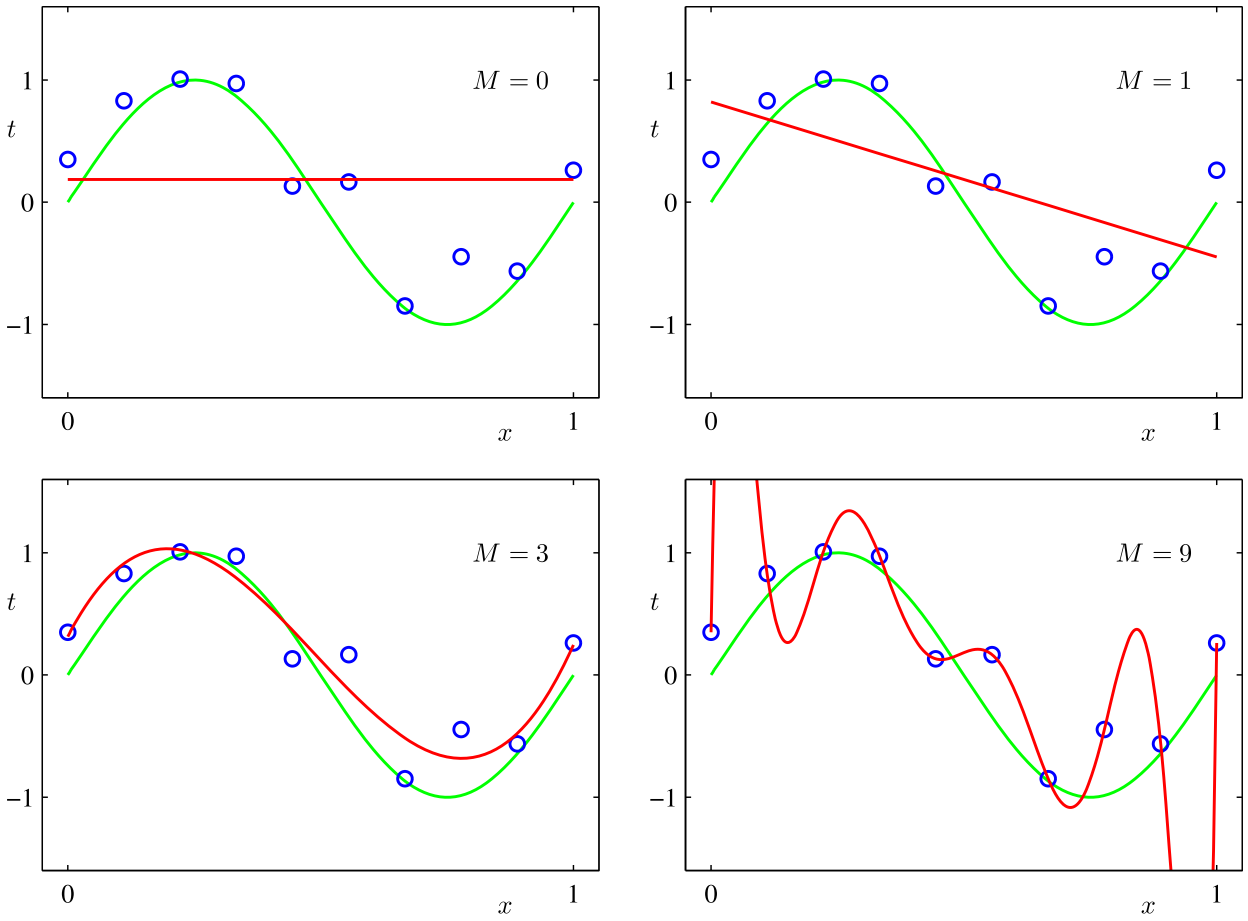


Figure 1.4 of Pattern Recognition and Machine Learning.

Linear Regression Example

To plot the error, the *root mean squared error* $\mathrm{RMSE} = \sqrt{\mathrm{MSE}}$ is frequently used.

The displayed error nicely illustrates two main challenges in machine learning:

- *underfitting*
- *overfitting*

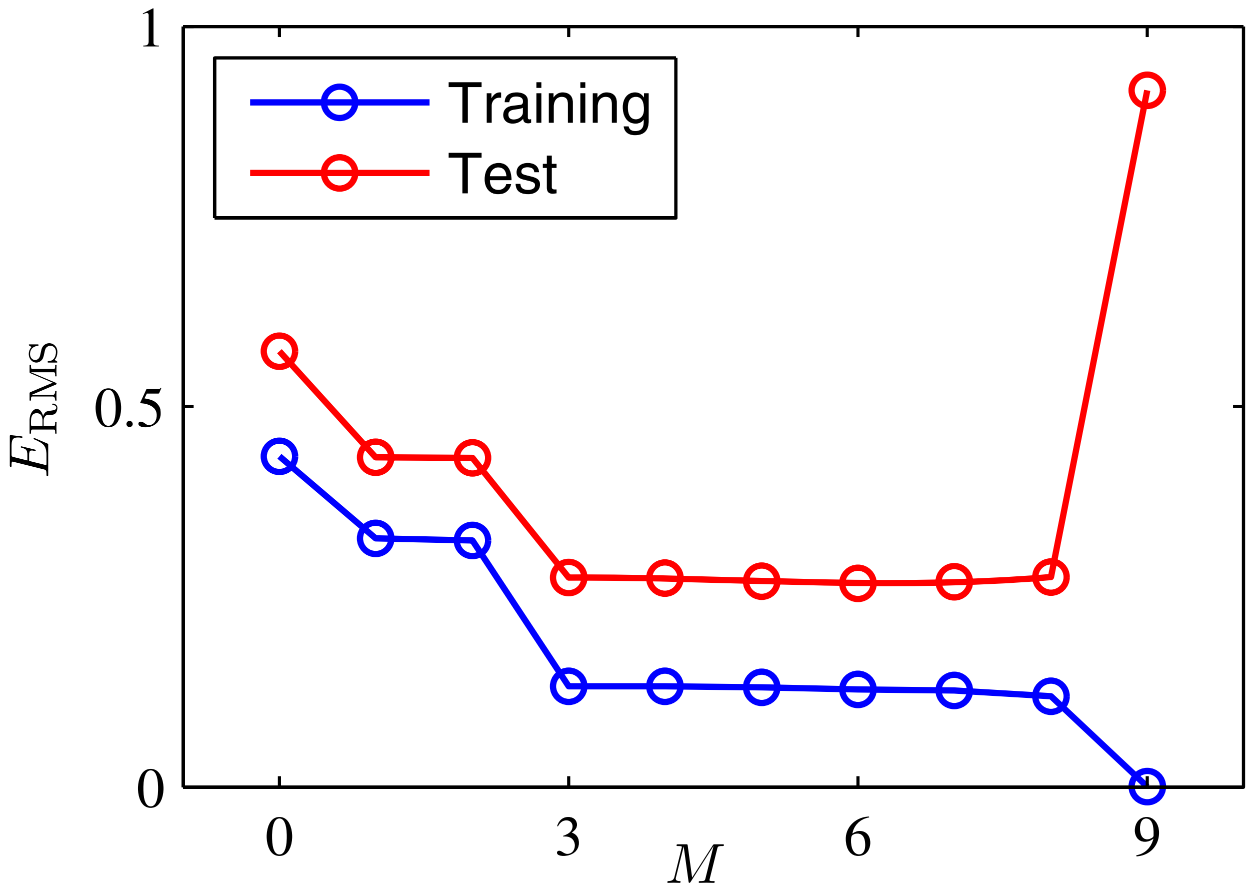


Figure 1.5 of Pattern Recognition and Machine Learning.

Model Capacity

We can control whether a model underfits or overfits by modifying its *capacity*.

- representational capacity
- effective capacity

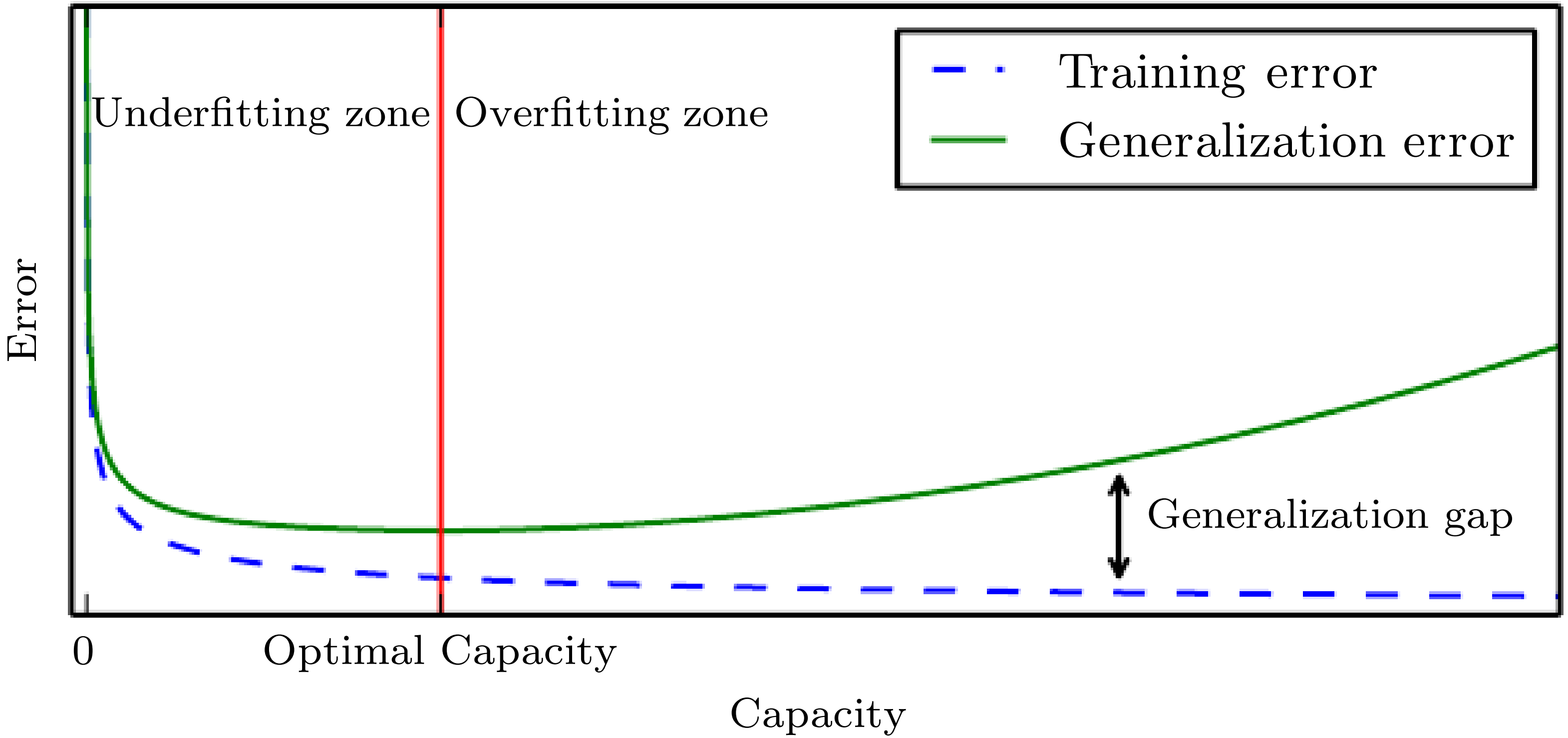


Figure 5.3, page 115 of Deep Learning Book, http://deeplearningbook.org

Linear Regression Overfitting

Note that employing more data also usually alleviates overfitting (the relative capacity of the model is decreased).

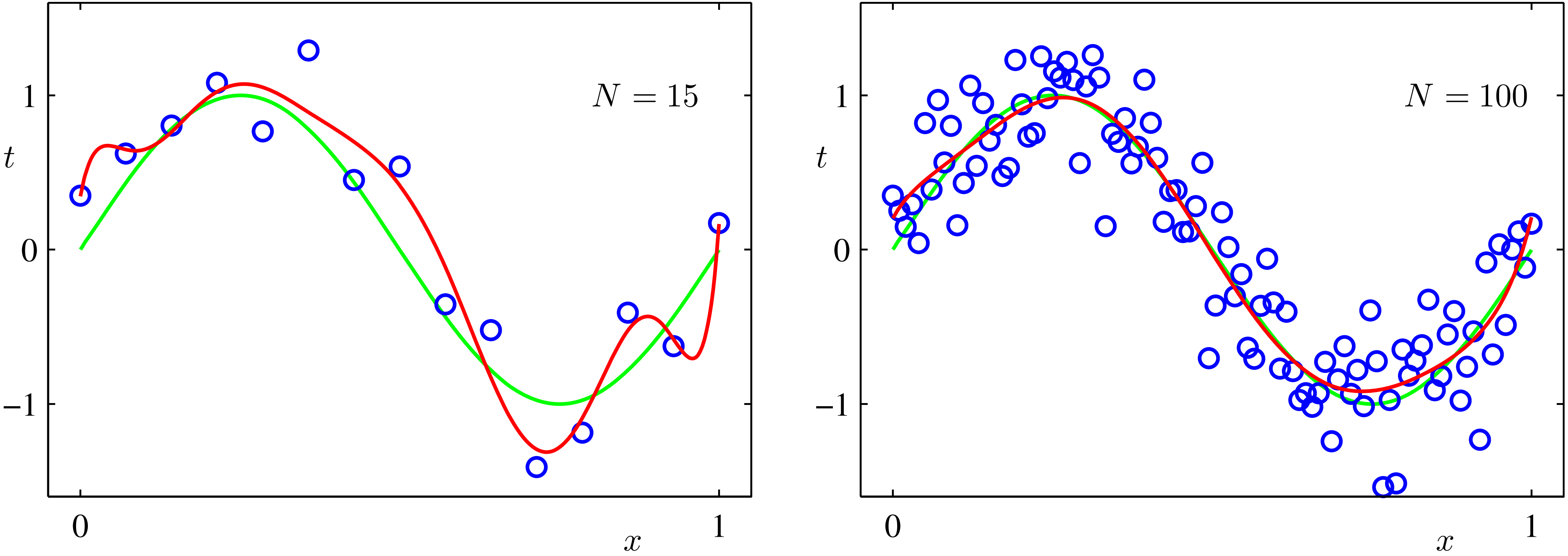


Figure 1.6 of Pattern Recognition and Machine Learning.

Regularization

Regularization in a broad sense is any change in a machine learning algorithm that is designed to *reduce generalization error* but not necessarily its training error).

L_2 regularization (also called weighted decay) penalizes models with large weights:

$$\frac{1}{2}\sum_{i=1}^{N}\left(f(\boldsymbol{x}_i;\boldsymbol{w})-t_i\right)^2+\frac{\lambda}{2}||\boldsymbol{w}||^2$$

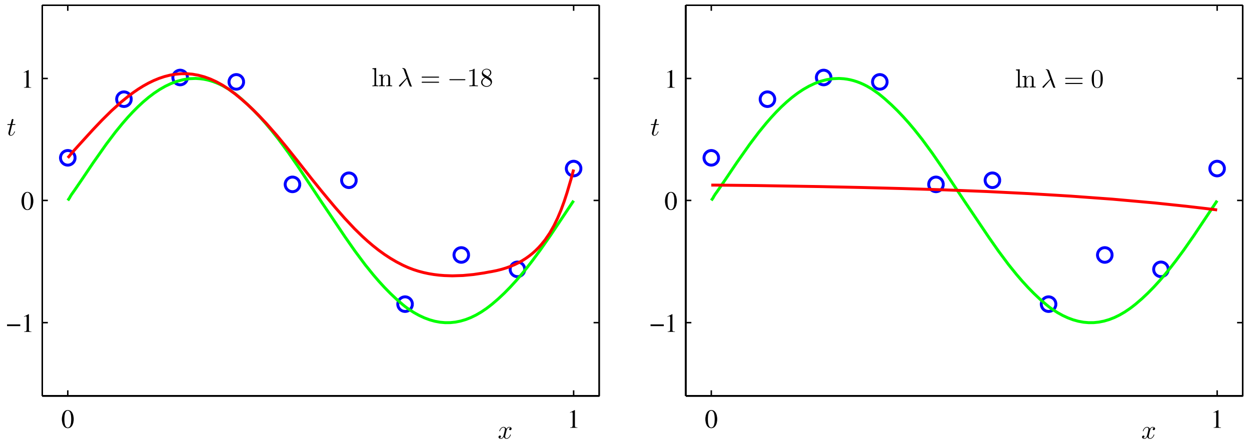


Figure 1.7 of Pattern Recognition and Machine Learning.

Regularizing Linear Regression

In matrix form, regularized sum of squares error for linear regression amounts to

$$\frac{1}{2}||\boldsymbol{X}\boldsymbol{w} - \boldsymbol{t}||^2 + \frac{\lambda}{2}||\boldsymbol{w}||^2.$$

When repeating the same calculation as in the unregularized case, we arrive at

$$(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{t},$$

where $\boldsymbol{I}$ is an identity matrix.

Input: Dataset ($\boldsymbol{X} \in \mathbb{R}^{N\times D}$, $\boldsymbol{t} \in \mathbb{R}^N$), constant $\lambda \in \mathbb{R}^+$.
Output: Weights $\boldsymbol{w} \in \mathbb{R}^D$ minimizing MSE of regularized linear regression.

- $\boldsymbol{w} \leftarrow (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{t}.$

Choosing Hyperparameters

Hyperparameters are not adapted by the learning algorithm itself.

Usually a **validation set** or **development set** is used to estimate the generalization error, allowing to update hyperparameters accordingly. If there is not enough data (well, there is **always** not enough data), more sophisticated approaches can be used.

So far, we have seen two hyperparameters, M and λ.

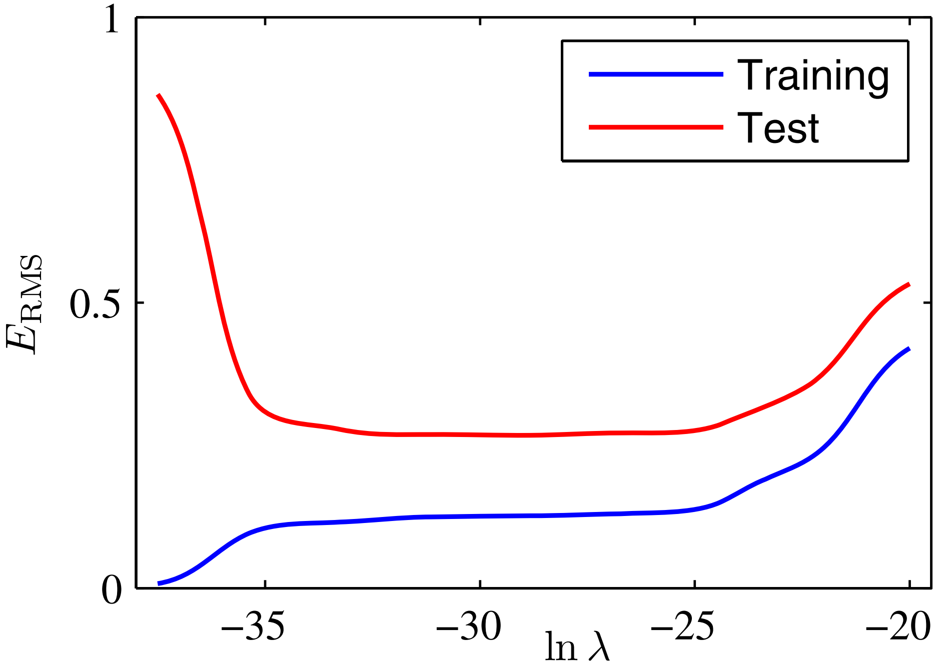


Figure 1.8 of Pattern Recognition and Machine Learning.