

Introduction to Machine Learning

Milan Straka

 October 04, 2021



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



unless otherwise stated

Course Website: <https://ufal.mff.cuni.cz/courses/npfl129>

- Slides, recordings, assignments, Exam questions

Course Repository: <https://github.com/ufal/npfl129>

- Templates for the assignments, slide sources.

Piazza

- Piazza will be used as a communication platform.

You can post questions or notes,

- privately to the instructors, or
- to everyone (signed or anonymously).

Students can answer other student's questions too, which allows you to get faster response. Please do not send complete solutions to other students, only excerpts of the source files.

- Please use Piazza for **all communication** with the instructors.
- You will get the invite link after the first lecture.

<https://recodex.mff.cuni.cz>

- The assignments will be evaluated automatically in ReCodEx.
- If you have a MFF SIS account, you will be able to create an account using your CAS credentials and should automatically see the right group.
- Otherwise, there will be **instructions** on **Piazza** how to get ReCodEx account (generally you will need to send me a message with several pieces of information and I will send it to ReCodEx administrators in batches).

Practicals

- There will be 1-3 assignments a week, each with a 2-week deadline.
 - There is also another week-long second deadline, but for less points.
- After solving the assignment, you get non-bonus points, and sometimes also bonus points.
- To pass the practicals, you need to get 80 non-bonus points. There will be assignments for at least 120 non-bonus points.
- If you get more than 80 points (be it bonus or non-bonus), they will be transferred to the exam (but at most 40 points are transferred).

Lecture

You need to pass a written exam.

- All questions are publicly listed on the course website.
- There are questions for 100 points in every exam, plus at most 40 surplus points from the practicals and plus at most 10 surplus points for **community work** (improving slides, ...).
- You need 60/75/90 points to pass with grade 3/2/1; 75 points for PhD students.

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, ...*
- Measure P
 - *accuracy, error rate, F-score, ...*
- 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, ...*

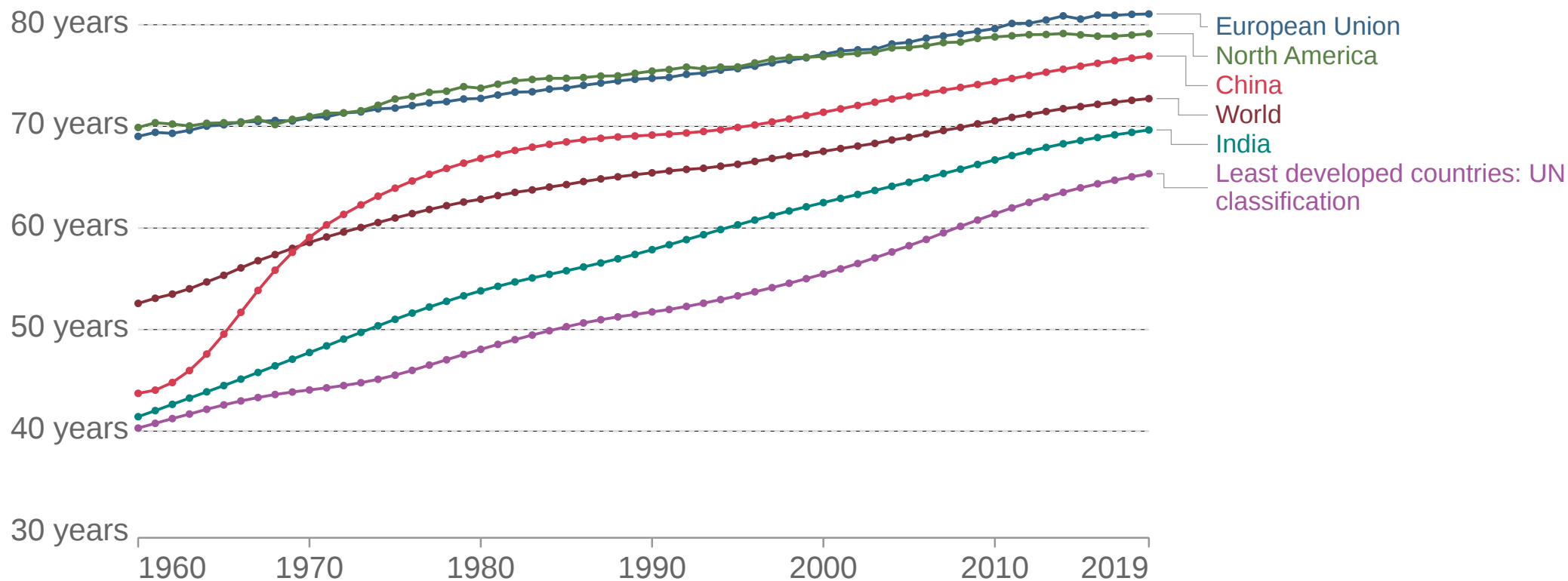


Image from "ImageNet Classification with Deep Convolutional Neural Networks" paper by Alex Krizhevsky et al.

Life expectancy

The average number of years a newborn would live if age-specific mortality rates in the current year were to stay the same throughout its life.

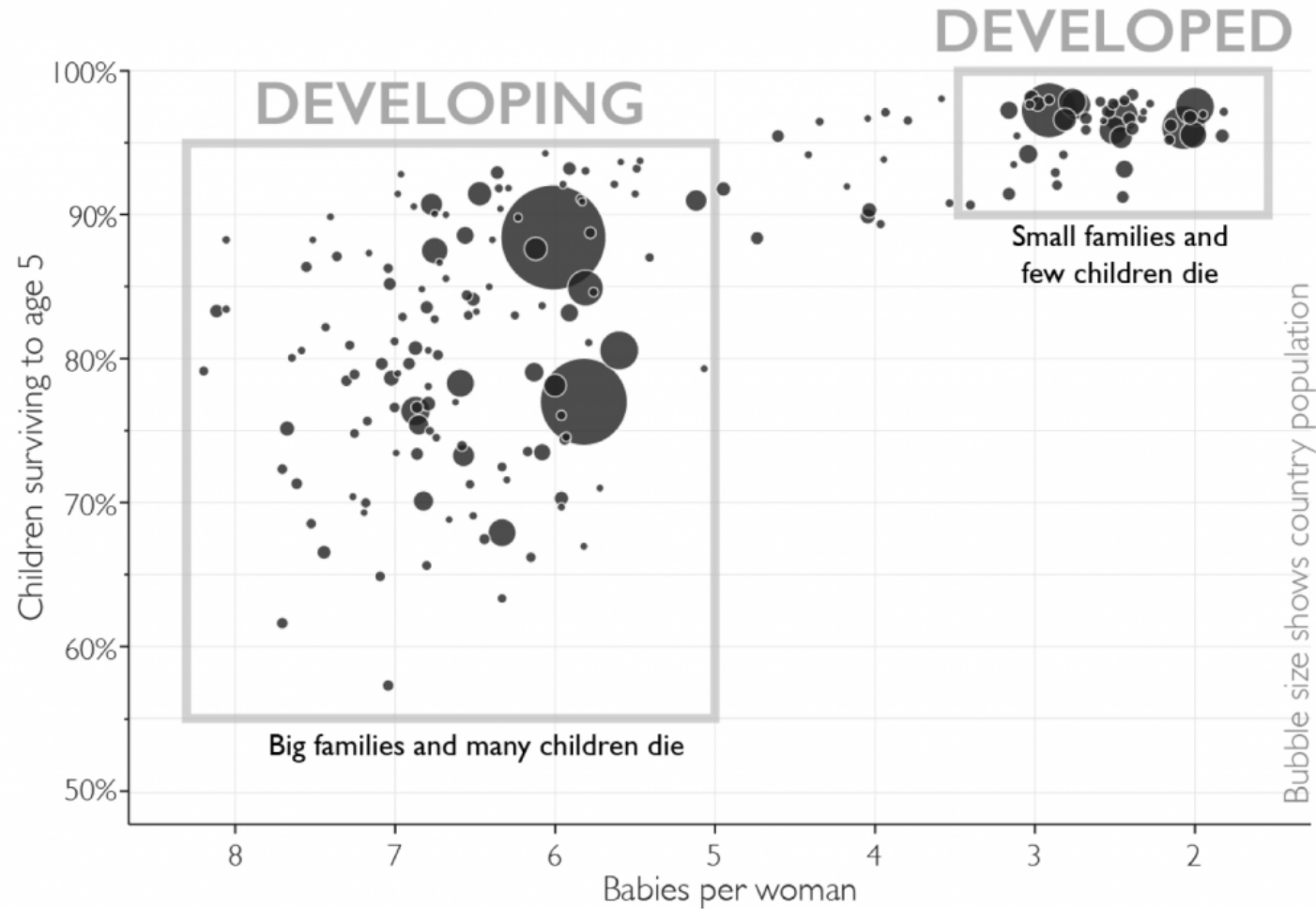
Our World
in Data



Source: Data compiled from multiple sources by World Bank

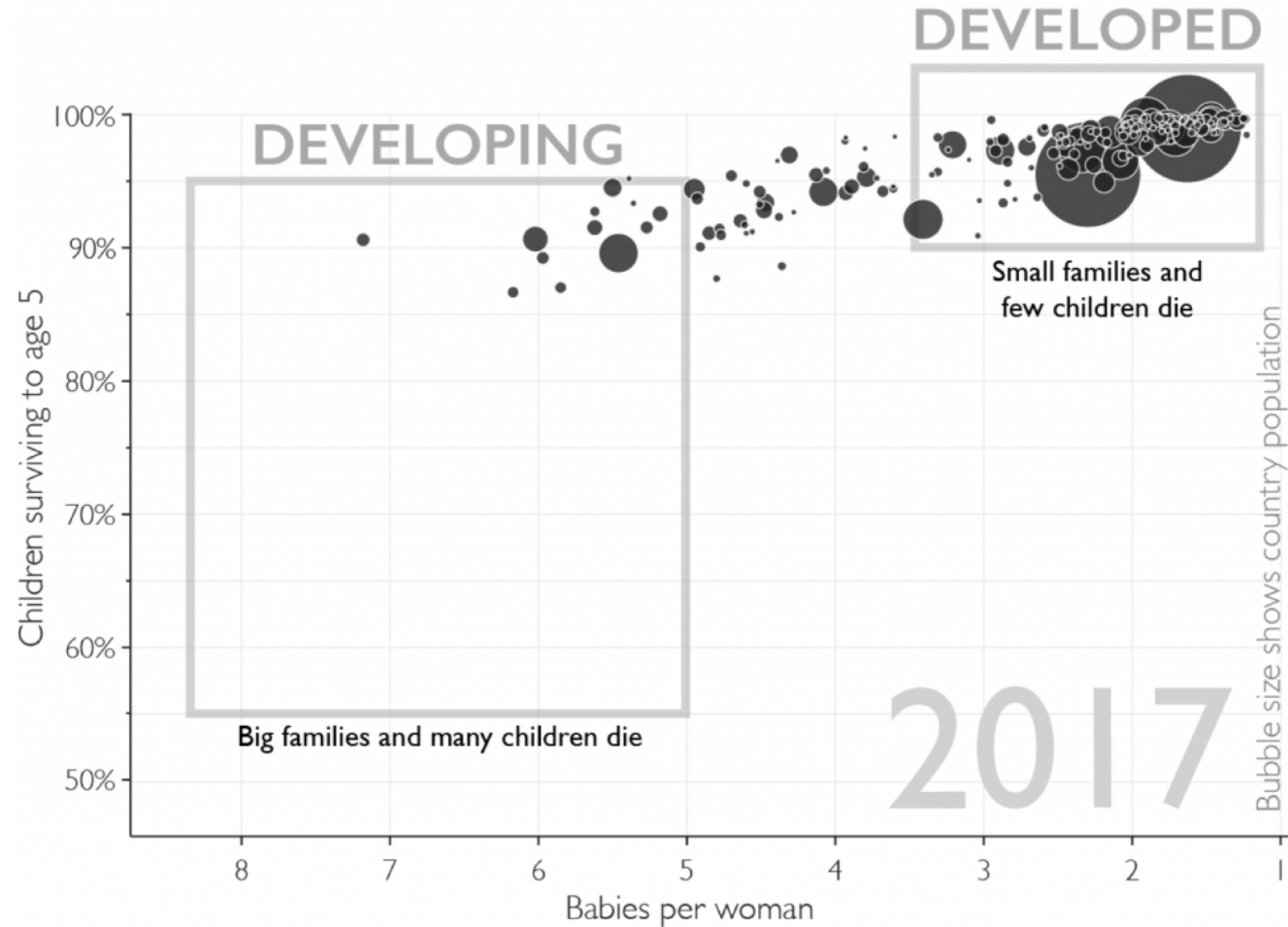
OurWorldInData.org/life-expectancy • CC BY

<https://ourworldindata.org/life-expectancy>



Sources: UN-IGME & UN-Pop[1,3]

<https://www.gapminder.org/topics/fertility-child-mortality/>

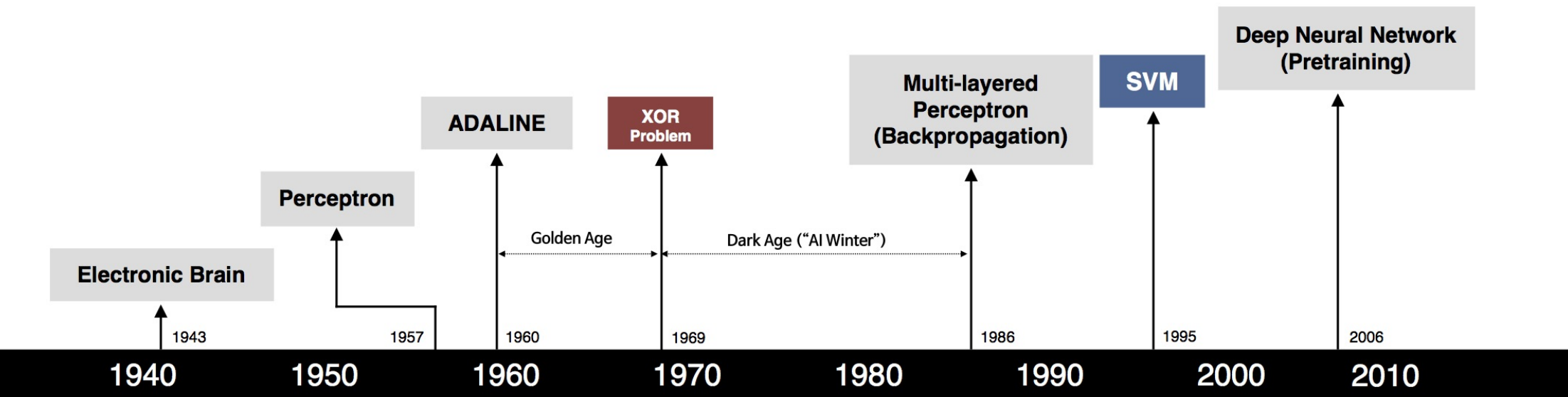


Sources: UN-IGME, UN-Pop[1,3] & Gapminder[6]

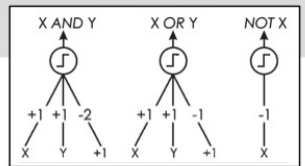
<https://www.gapminder.org/topics/fertility-child-mortality/>

- 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
- StarCraft II

Introduction to Machine Learning History



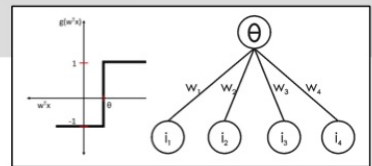
S. McCulloch – W. Pitts



- Adjustable Weights
- Weights are not Learned



F. Rosenblatt



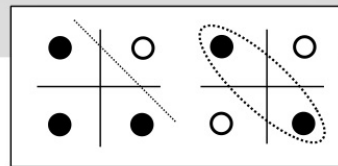
- Learnable Weights and Threshold



B. Widrow – M. Hoff



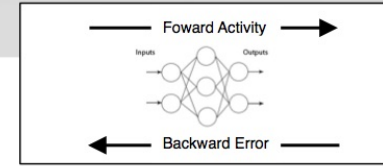
M. Minsky – S. Papert



- XOR Problem



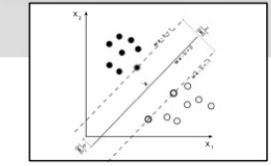
D. Rumelhart – G. Hinton – R. Williams



- Solution to nonlinearly separable problems
- Big computation, local optima and overfitting



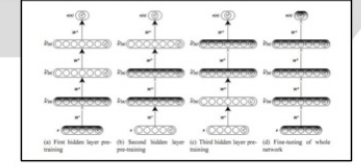
V. Vapnik – C. Cortes



- Limitations of learning prior knowledge
- Kernel function: Human Intervention



G. Hinton – S. Ruslan



- Hierarchical feature Learning

Assume we have an input of $\mathbf{x} \in \mathbb{R}^D$. The two basic ML tasks are:

1. **regression**: The goal of a regression is to predict real-valued target variable $t \in \mathbb{R}$ for 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 (\mathbf{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.

- a , \mathbf{a} , \mathbf{A} , \mathbf{A} : scalar (integer or real), vector, matrix, tensor
 - all vectors are always **column** vectors
 - transposition changes a column vector into a row vector, so \mathbf{a}^T is a row vector
 - we denote **scalar product** between vectors \mathbf{a} and \mathbf{b} as $\mathbf{a}^T \mathbf{b}$
 - we understand it as matrix multiplication
- \mathbf{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
- $\nabla_{\mathbf{x}} f(\mathbf{x})$: gradient of f with respect to \mathbf{x} , i.e., $\left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \dots, \frac{\partial f(\mathbf{x})}{\partial x_n} \right)$

Example Dataset

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

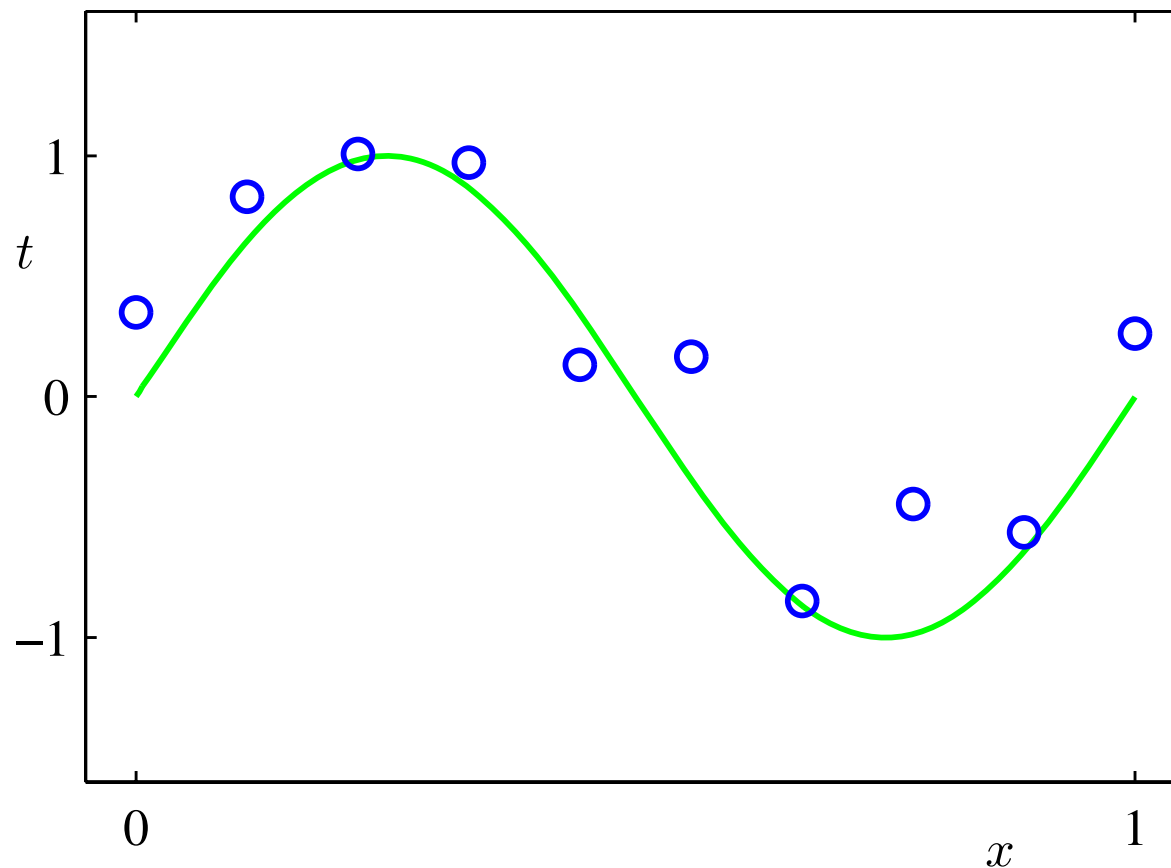


Figure 1.2 of Pattern Recognition and Machine Learning.

Usually, our machine learning algorithms will be trained using the **train set** $\mathbf{X} \in \mathbb{R}^{N \times D}$, which is a collection of N instances, each represented by D real numbers.

In supervised learning, we also have a **target** \mathbf{t} for every instance,

- a real number for regression, $\mathbf{t} \in \mathbb{R}^N$;
- a class for classification, $\mathbf{t} \in \{0, 1, \dots, K - 1\}^N$.

The input to machine learning algorithms is frequently preprocessed, i.e., the algorithms do not always work directly on the input \mathbf{X} , but on some modification of it. These preprocessed input values are called **features**.

In literature, the collection of the processed inputs is called a **design matrix** $\Phi \in \mathbb{R}^{N \times M}$.

However, we will denote the inputs to algorithms always as \mathbf{X} , be it the original training data or processed features.

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

$$y(\mathbf{x}; \mathbf{w}, b) = x_1 w_1 + x_2 w_2 + \dots + x_D w_D + b = \sum_{i=1}^D x_i w_i + b = \mathbf{x}^T \mathbf{w} + b.$$

The \mathbf{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 \mathbf{x} by padding a value 1, and consider only $\mathbf{x}^T \mathbf{w}$, 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 \mathbf{X} with Ones

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

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

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

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

Linear Regression

Assume we have a dataset of N input values $\mathbf{x}_1, \dots, \mathbf{x}_N$ and targets t_1, \dots, 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*:

$$\text{MSE}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (y(\mathbf{x}_i; \mathbf{w}) - t_i)^2.$$

Often, *sum of squares*

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

is used instead, because minimizing it is equal to minimizing MSE, but the math comes out nicer.

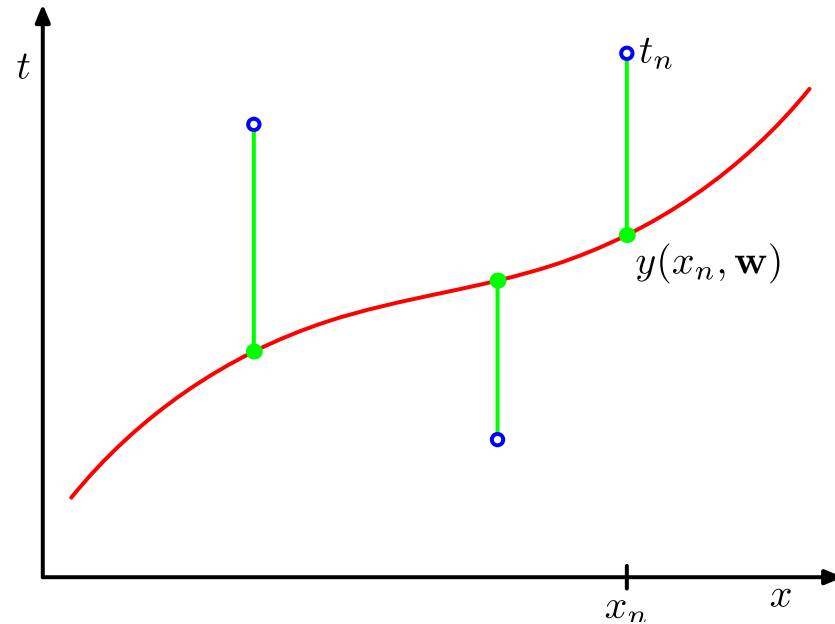


Figure 1.3 of Pattern Recognition and Machine Learning.

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:

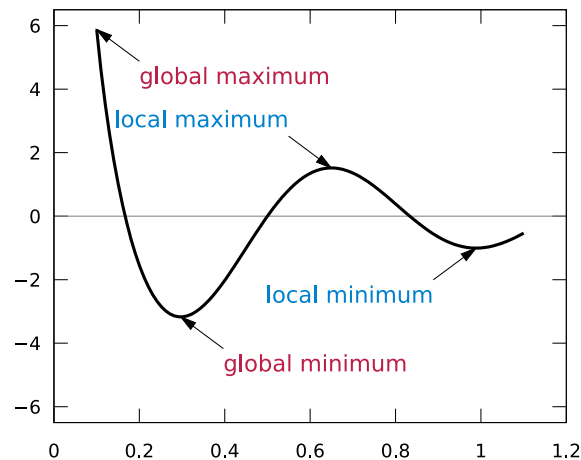
$$\frac{1}{2} \sum_i^N (\mathbf{x}_i^T \mathbf{w} - t_i)^2.$$

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

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

Minimization – Unconstrained, Single Real Variable

Assume we have a function and we want to find its minimum.



https://commons.wikimedia.org/wiki/File:Extrema_example_original.svg

We usually use the Fermat's theorem (interior extremum theorem):

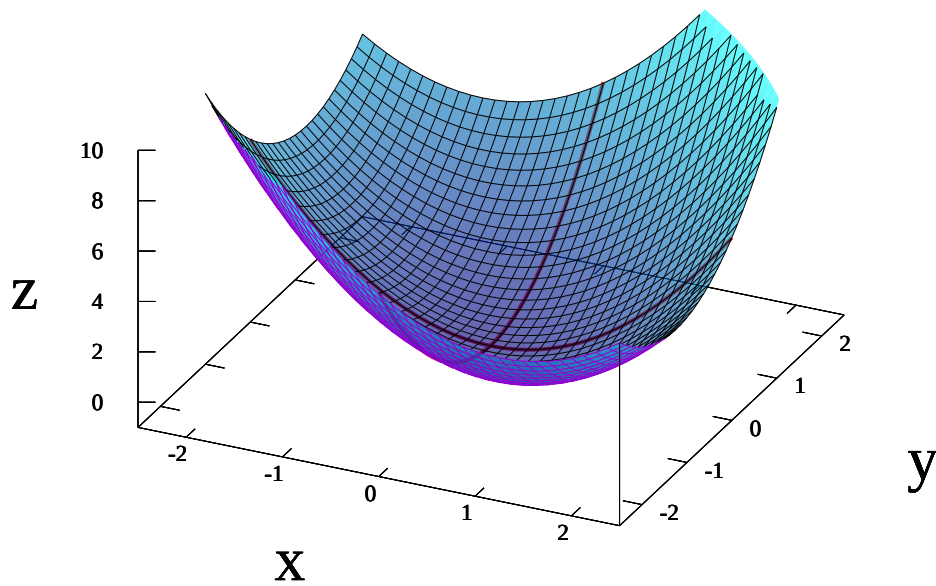
Let $f : \mathbb{R} \rightarrow \mathbb{R}$ be a function. If it has minimum (or maximum) in x and if it has derivative in x , then

$$\frac{\partial f}{\partial x} = 0.$$

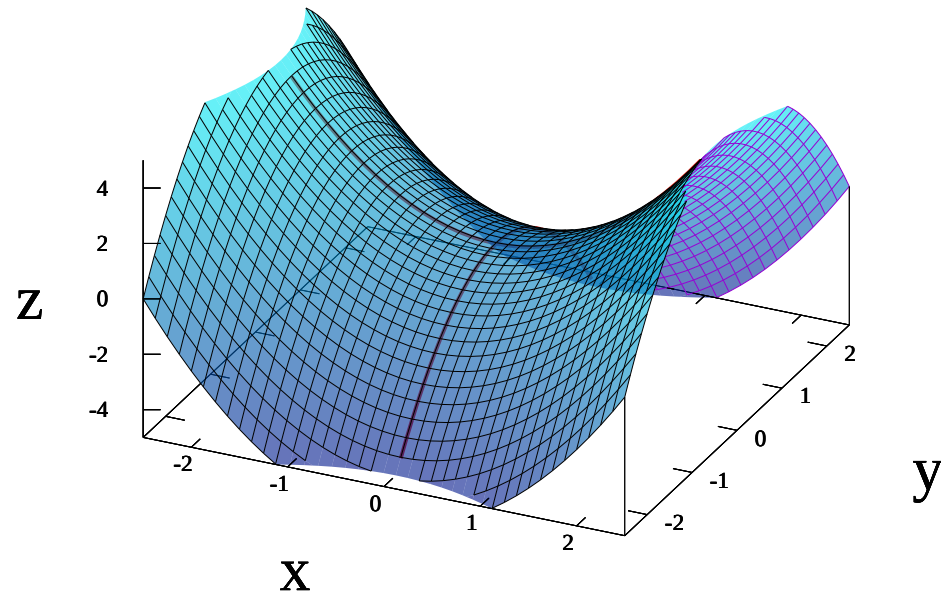
Minimization – Unconstrained, Multiple Real Variables

The previous theorem can be generalized to the multivariate case:

Let $f : \mathbb{R}^D \rightarrow \mathbb{R}$ be a function. If it has minimum (or maximum) in $\mathbf{x} = (x_1, x_2, \dots, x_D)$ and if it has derivative in \mathbf{x} , then for all i , $\frac{\partial f}{\partial x_i} = 0$. In other words, $\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$.



https://commons.wikimedia.org/wiki/File:Partial_func_eg.svg



https://commons.wikimedia.org/wiki/File:Partial_func_eg.svg

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

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

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

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

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

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

Input: Dataset $(\mathbf{X} \in \mathbb{R}^{N \times D}, \mathbf{t} \in \mathbb{R}^N)$.

Output: Weights $\mathbf{w} \in \mathbb{R}^D$ minimizing MSE of linear regression.

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

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

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

Linear Regression Example

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

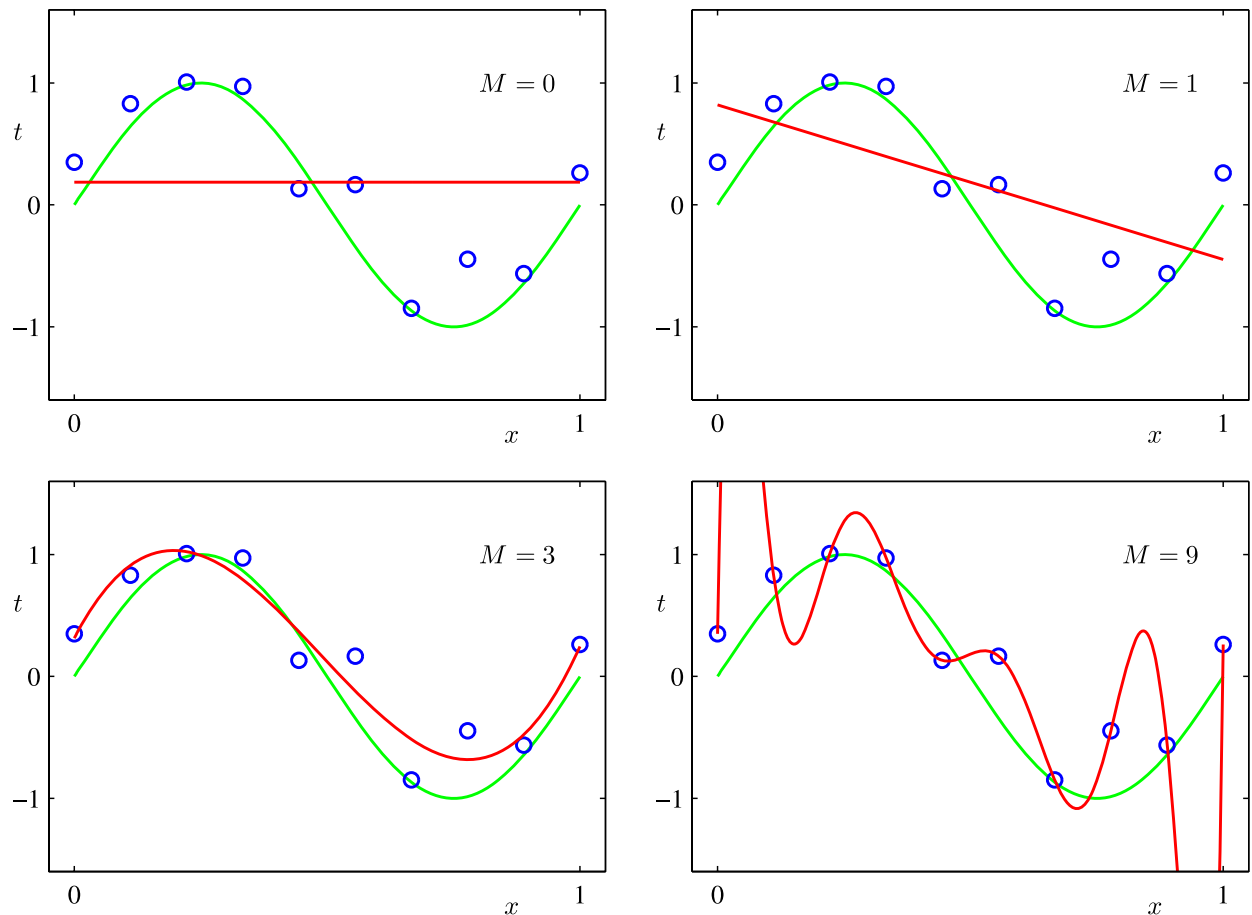


Figure 1.4 of *Pattern Recognition and Machine Learning*.

Linear Regression Example

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

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

- *underfitting*
- *overfitting*

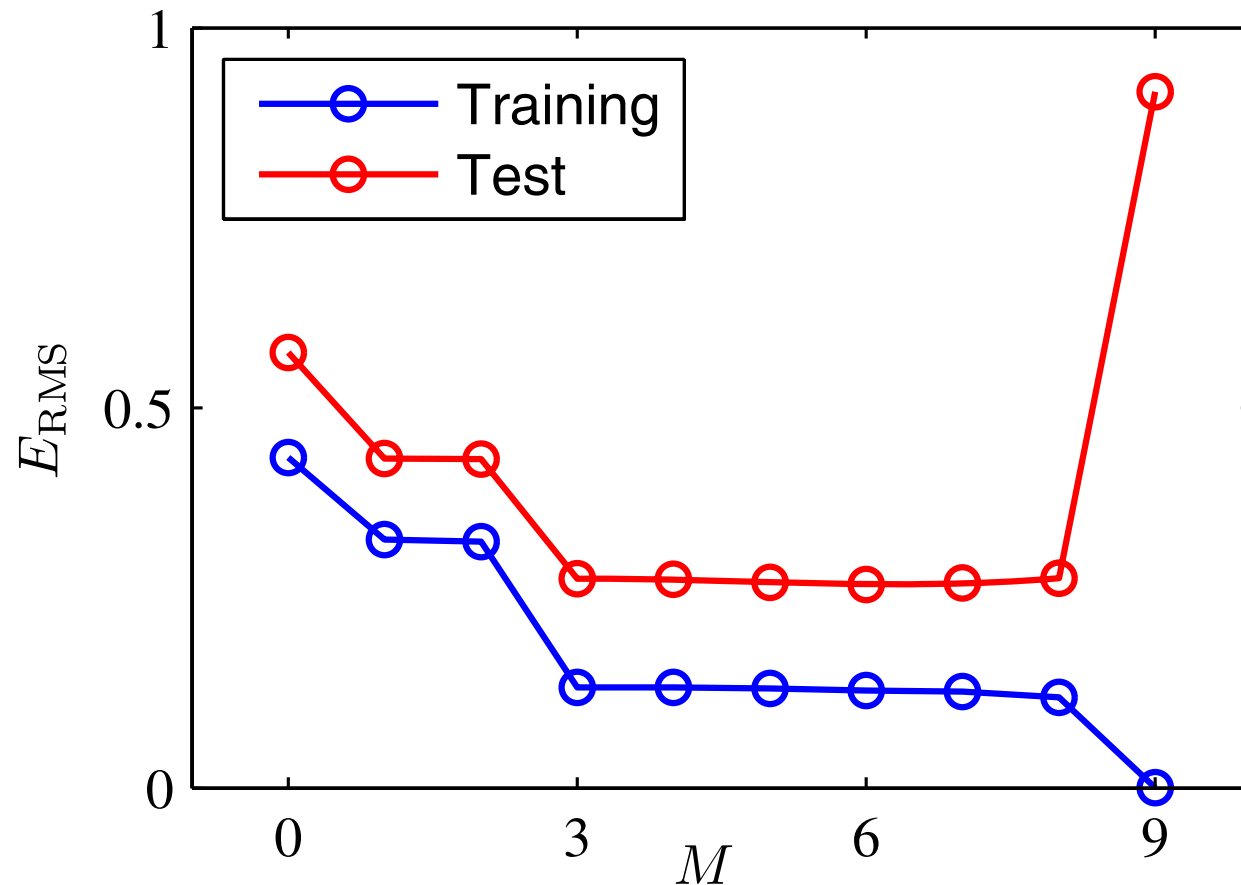


Figure 1.5 of Pattern Recognition and Machine Learning.