Introduction to Machine Learning

Milan Straka

October 05, 2020

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

Organization

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

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

Zoom

- The lectures and practicals are happening on Zoom.
- The recordings will be available from the course website.

Piazza

- Piazza will be used as a communication platform. It allows sending
  - either notes or questions (the latter require an answer)
  - to everybody (signed or anonymously), to all instructors, to a specific instructor
  - students can answer other students' questions too
- Please use it whenever possible for communication with the instructors.
- You will get the invite link after the first lecture.

ReCodEx

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 will be automatically assigned to the right group.
- Otherwise follow the instructions on Piazza; generally you will need to send me a message with several pieces of information and I will send it to ReCodEx administrators in batches.

Course Requirements

Practicals

- There will be 1-2 assignments a week, each with 2-week deadline.
  - Deadlines can be extended, but you need to write **before** the deadline.
- 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 (e.g., improving slides).
- You need 60/75/90 points to pass with grade 3/2/1.

Machine Learning

A possible definition of learning from Mitchell (1997):

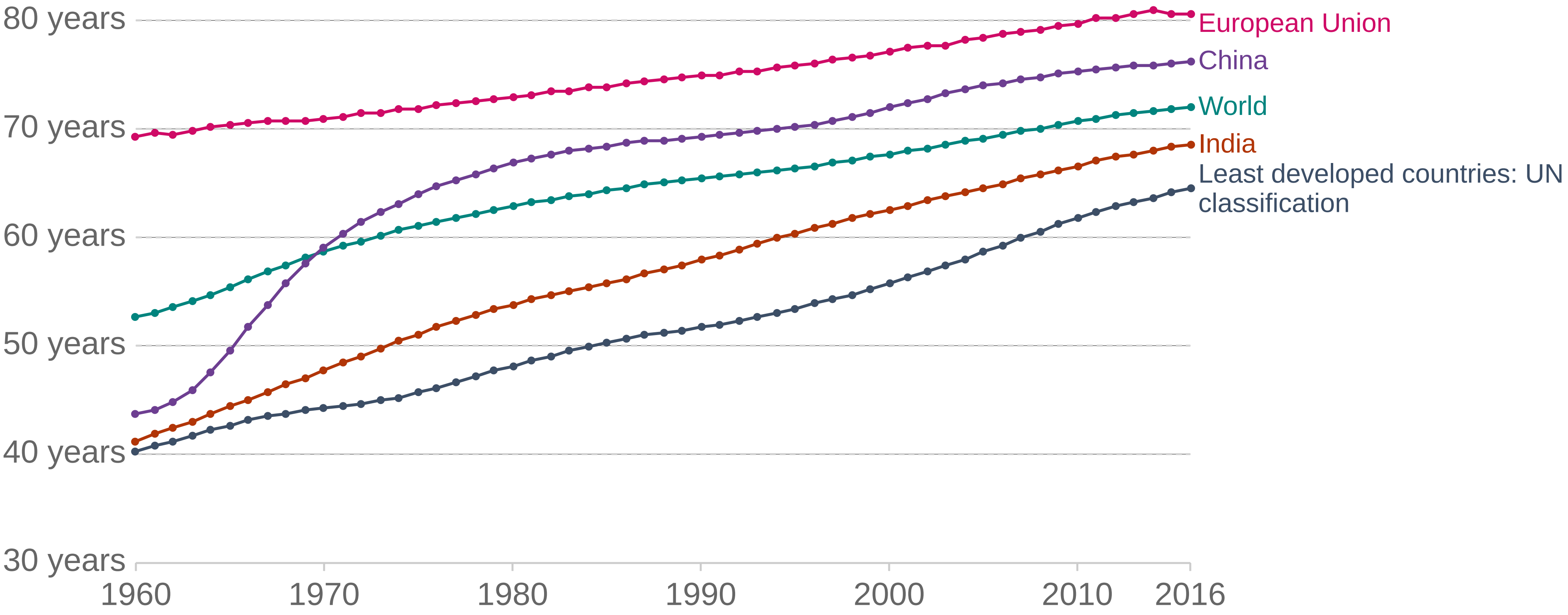
> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

- 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*, …
- 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*, …

Supervised Machine Learning

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.

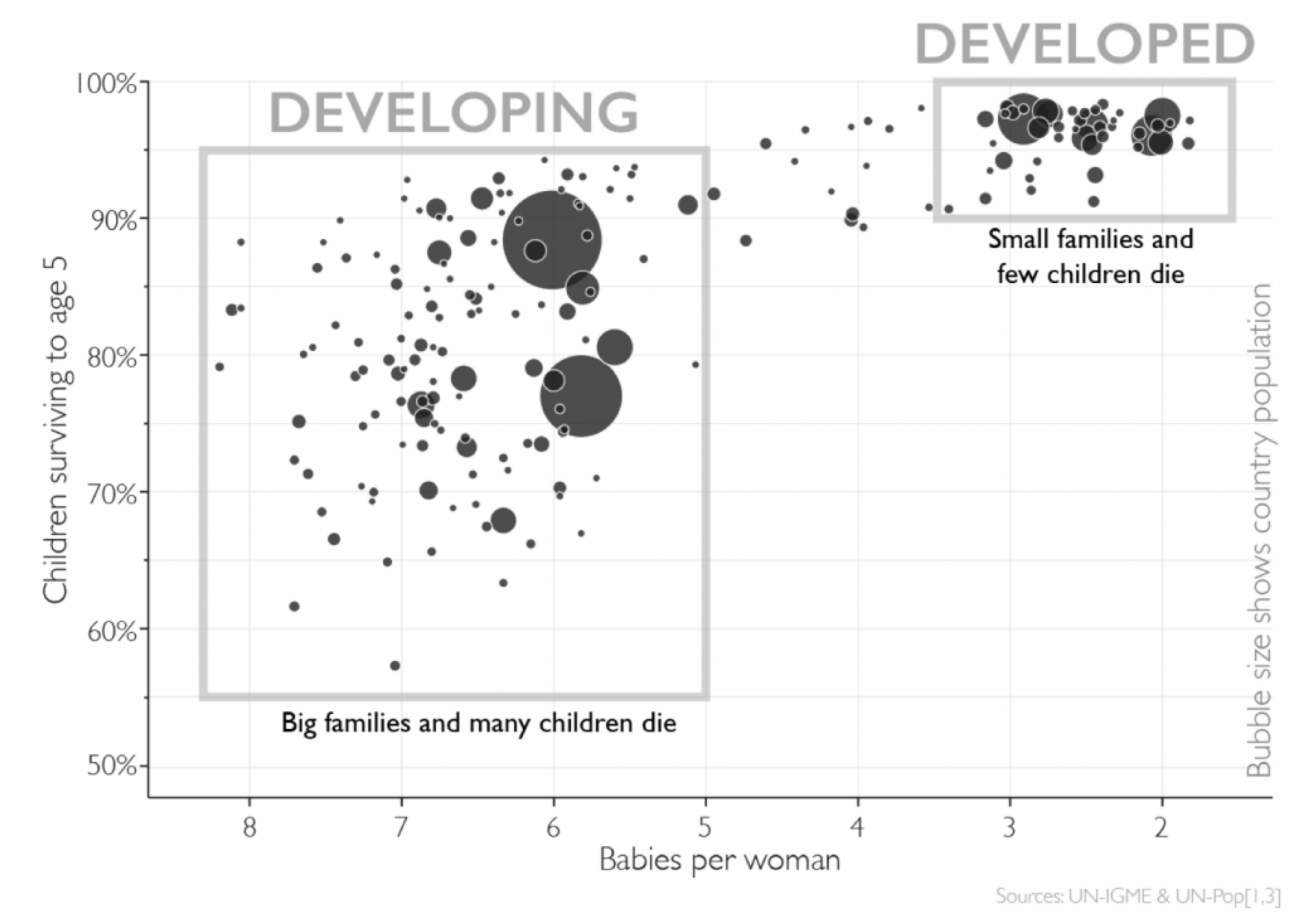


Source: UN Population Division; World Bank

OurWorldInData.org/life-expectancy • CC BY

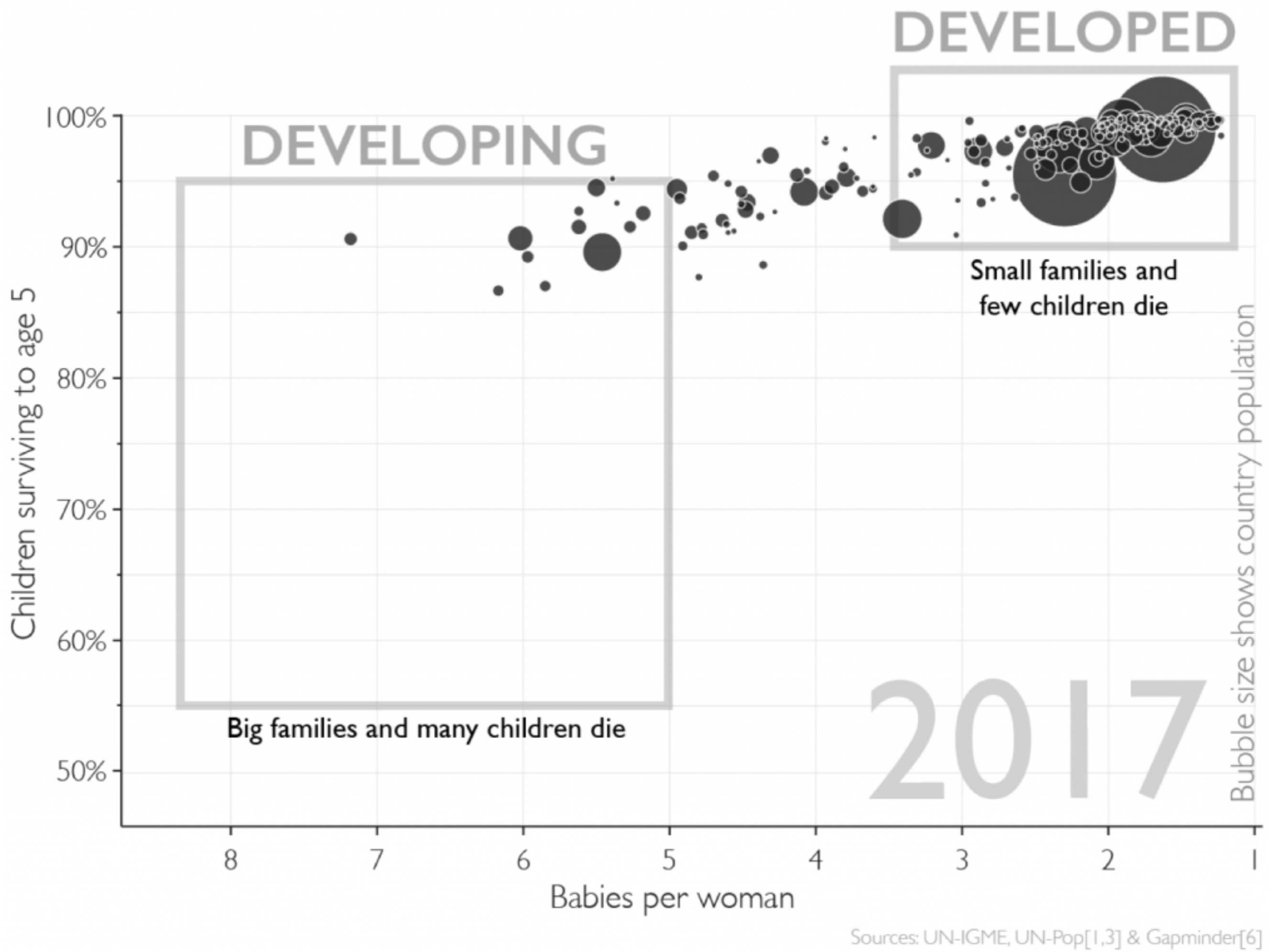
https://ourworldindata.org/life-expectancy

Unsupervised Machine Learning – Clustering



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

Unsupervised Machine Learning – Clustering

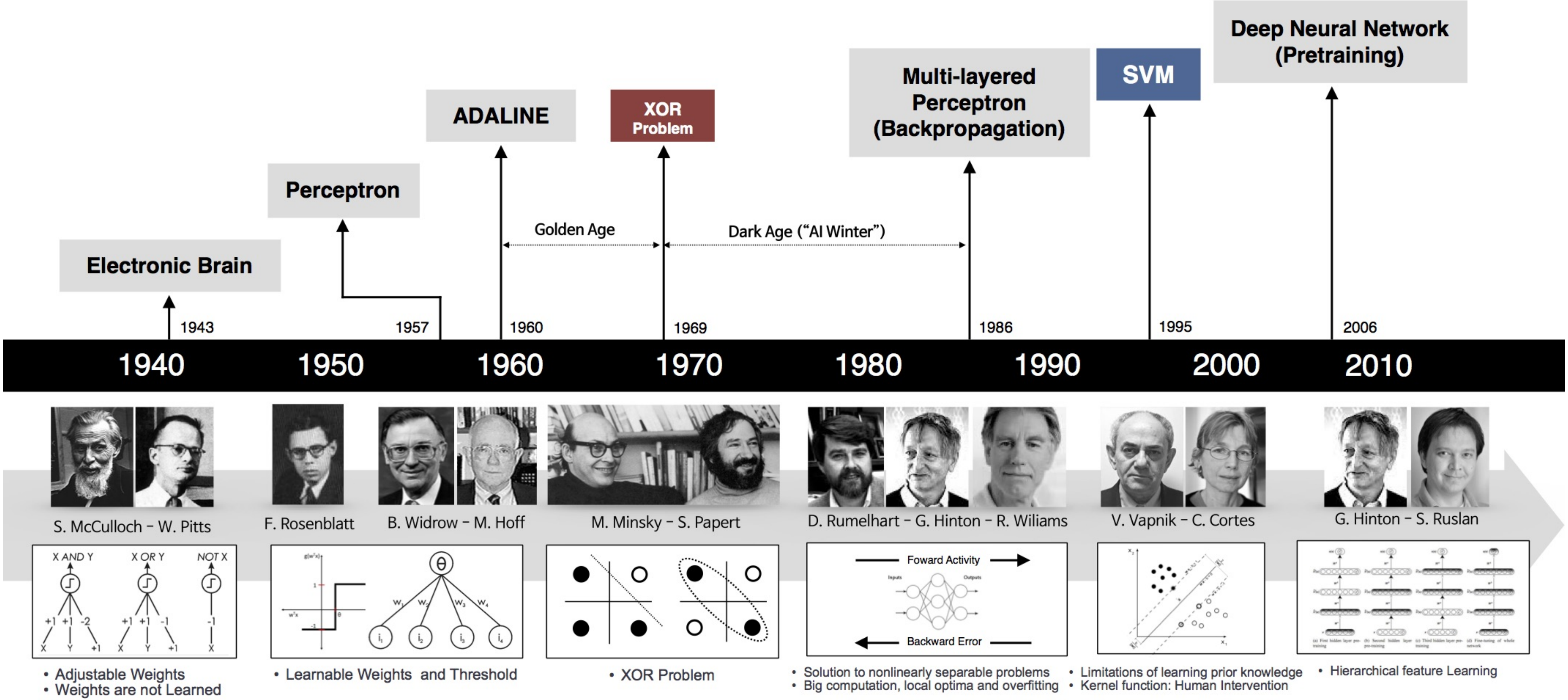


https://www.gapminder.org/topics/fertility-child-mortality/

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
- StarCraft II

Introduction to Machine Learning History



Electronic Brain — 1943

Perceptron — 1957

ADALINE — 1960

XOR Problem — 1969

Multi-layered Perceptron (Backpropagation) — 1986

SVM — 1995

Deep Neural Network (Pretraining) — 2006

Golden Age

Dark Age (“AI Winter”)

1940 1950 1960 1970 1980 1990 2000 2010

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. Wiliams

- 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

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

Machine and Representation Learning

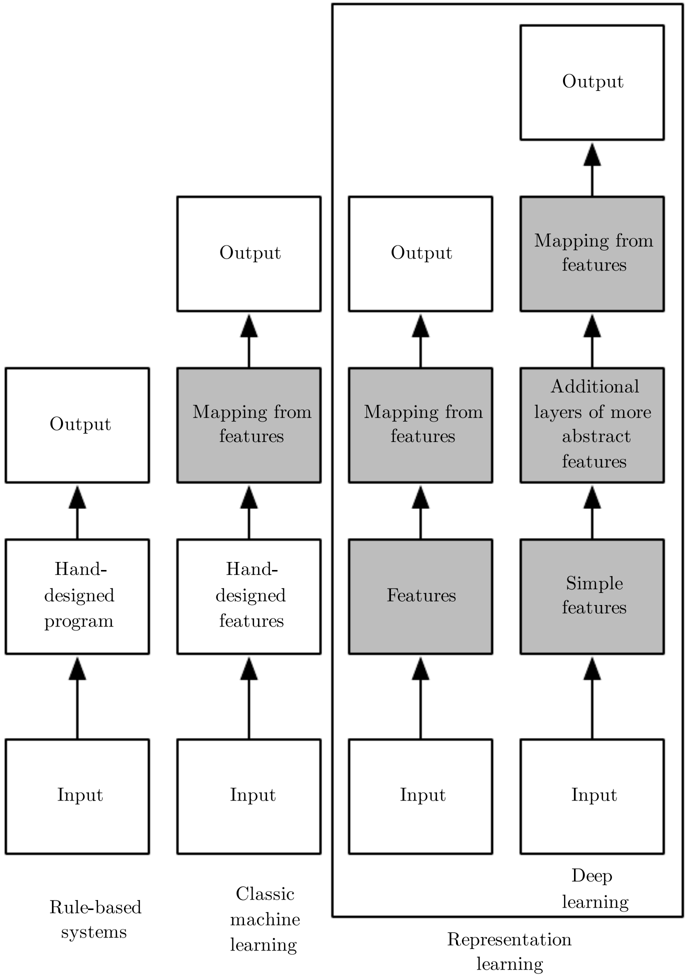


Figure 1.5, page 10 of Deep Learning Book, http://deeplearningbook.org.

Basic Machine Learning Settings

Assume we have an input of $\boldsymbol{x} \in \mathbb{R}^D$. 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 $(\boldsymbol{x}, t)$ 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 noise.

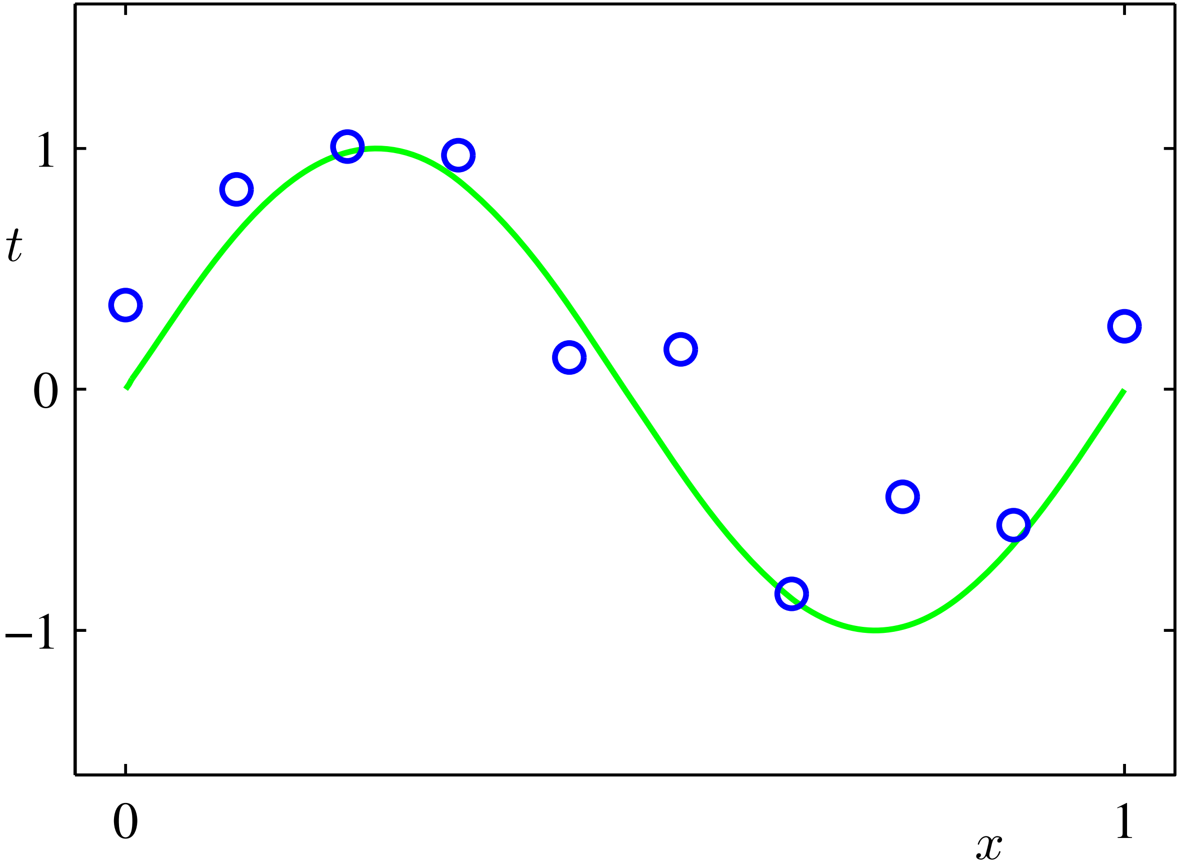


Figure 1.2 of Pattern Recognition and Machine Learning.

Input Data

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

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

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

The input to machine learning algorithms is frequently preprocessed, i.e., the algorithms do not always work directly on the input $\boldsymbol{X}$, 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** $\boldsymbol{\Phi} \in \mathbb{R}^{N \times M}$. However, we will denote the inputs to algorithms always $\boldsymbol{X}$, be it the original training data or processed features.

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**:

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

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 $y(x) = \boldsymbol{x}^T\boldsymbol{w} + b$.

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

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

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

Linear Regression

Assume we have a dataset of N input values $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ 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*:

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

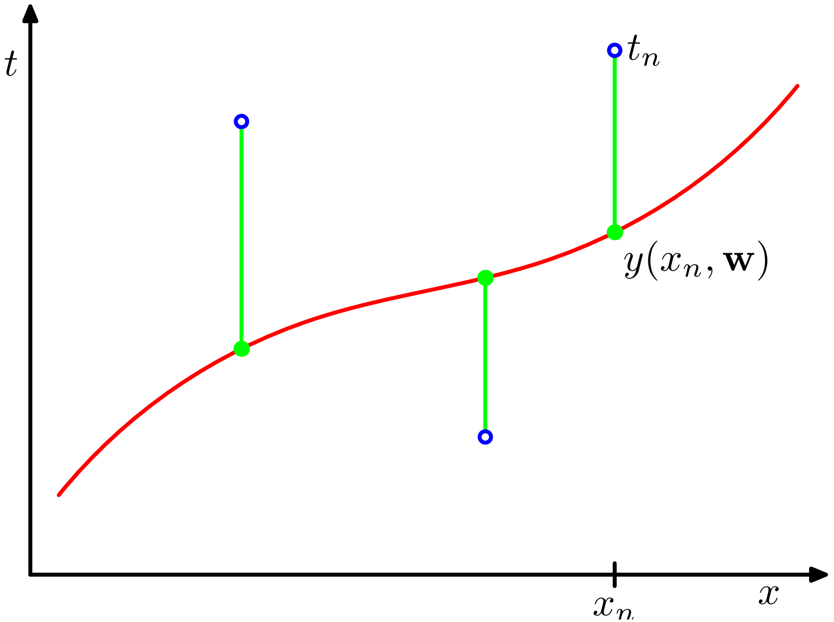


Figure 1.3 of Pattern Recognition and Machine Learning.

Often, *sum of squares*

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

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

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:

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

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.

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

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 $M \geq 0$.

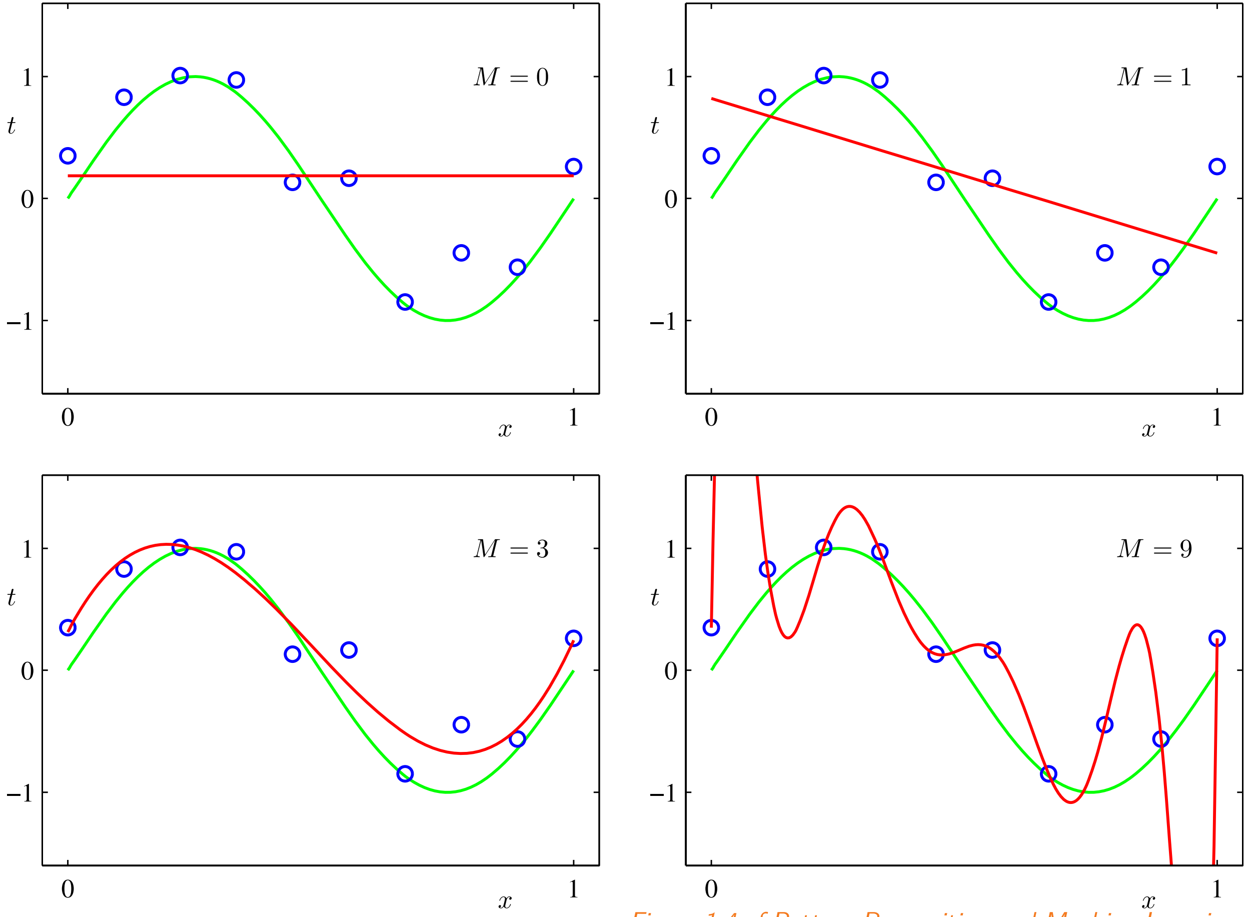


Figure 1.4 of Pattern Recognition and Machine Learning.

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*

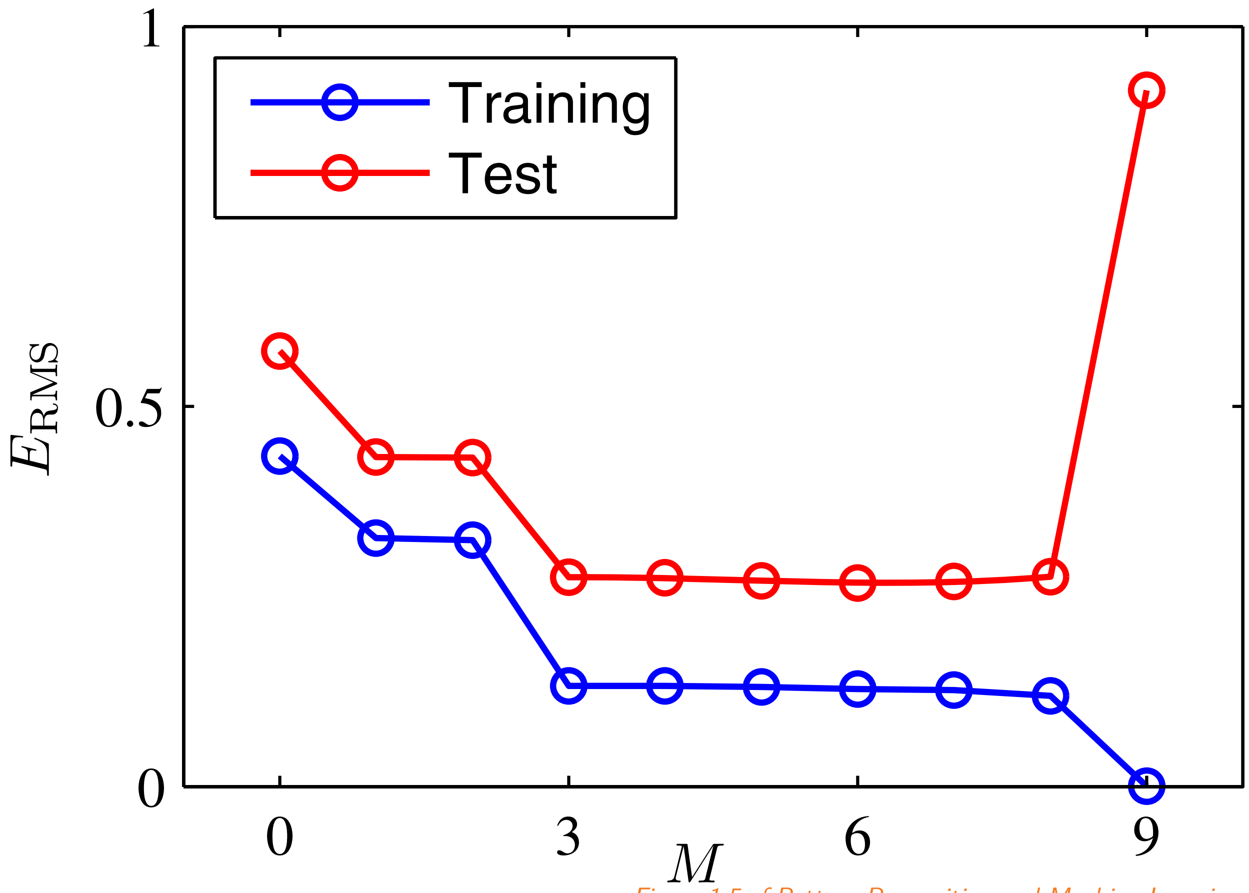


Figure 1.5 of Pattern Recognition and Machine Learning.