technische universität
dortmund

DeepLearning on FPGAs

Introduction to Data Mining

Sebastian Buschjäger

Technische Universität Dortmund - Fakultät Informatik - Lehrstuhl 8

October 11, 2017

Structure of this course

Goals

→ Learning the basics of Data Mining
→ Learning the basics of Deep Learning
→ Learning the basics of FPGA programming

[1] https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/

Structure of this course

Goals

→ Learning the basics of Data Mining
→ Learning the basics of Deep Learning
→ Learning the basics of FPGA programming

Small lecture-phase in the beginning

- **Week 1 - 3:** Data Mining and Deep Learning
- **Week 4 - 5:** FPGAs and Software

[1] https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/

Structure of this course

Goals

→ Learning the basics of Data Mining
→ Learning the basics of Deep Learning
→ Learning the basics of FPGA programming

Small lecture-phase in the beginning

- **Week 1 - 3:** Data Mining and Deep Learning
- **Week 4 - 5:** FPGAs and Software

Goal Dogs vs. Cats Kaggle competition[1]

- Image classification on FPGA with Deep Learning
- Train classifier on FPGA with Deep Learning

[1] https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/

The Goal: Predict dogs and cats

Overall Computer Science Approach

Technical Problem	Mathematical problem formulation
↓	
Mathematical Method	Theoretical method to solve problem
↓	
Algorithm	Theoretical algorithm implementing method
↓	
Implementation	Actual implementation

Overall Computer Science Approach: Example

Technical Problem	Best route from v_s to n_e in graph
↓	
Mathematical Method	Theoretical method to solve problem
↓	
Algorithm	Theoretical algorithm implementing method
↓	
Implementation	Actual implementation

Overall Computer Science Approach: Example

Technical Problem	Best route from v_s to n_e in graph
↓	
Mathematical Method	Single source shortest path problem
↓	
Algorithm	Theoretical algorithm implementing method
↓	
Implementation	Actual implementation

Overall Computer Science Approach: Example

Technical Problem	Best route from v_s to n_e in graph
↓	
Mathematical Method	Single source shortest path problem
↓	
Algorithm	Dijkstra, A*, Floyd-Warhsall, ...
↓	
Implementation	Actual implementation

Overall Computer Science Approach: Example

Technical Problem	Best route from v_s to n_e in graph
↓	
Mathematical Method	Single source shortest path problem
↓	
Algorithm	Dijkstra, A*, Floyd-Warhsall, ...
↓	
Implementation	`C,Java,Python,...`

Data Mining Basics

What is Data Mining?

Data Mining Basics

"The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use."

Data Mining Basics

"The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use."

Fact: Data Mining follows the same general approach
But: Some problems are hard to be exactly formalised and thus need some special treatment

Data Mining Basics

"The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use."

Fact: Data Mining follows the same general approach
But: Some problems are hard to be exactly formalised and thus need some special treatment

Example: Find all cats on the given pictures
$\rightarrow$ What is a mathematical representation of a cat?

Data Mining Basics

"The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use."

Fact: Data Mining follows the same general approach
But: Some problems are hard to be exactly formalised and thus need some special treatment

Example: Find all cats on the given pictures
$\rightarrow$ What is a mathematical representation of a cat?

Idea: Formalise given problem by positive and negative examples
$\rightarrow$ That is our data

Data Mining Basics

Problem 1: Data needs to be gathered and pre-processed
$\rightarrow$ crawling the web for images with tag "cat"

Data Mining Basics

Problem 1: Data needs to be gathered and pre-processed
$\rightarrow$ crawling the web for images with tag "cat"
Problem 2: Totally unclear what knowledge our data might contain
$\rightarrow$ cats and dogs can be on the same picture
$\Rightarrow$ We have to "mine" data and knowledge from it

Data Mining Basics

Problem 1: Data needs to be gathered and pre-processed
$\rightarrow$ crawling the web for images with tag "cat"
Problem 2: Totally unclear what knowledge our data might contain
$\rightarrow$ cats and dogs can be on the same picture
$\Rightarrow$ We have to "mine" data and knowledge from it

Data Mining is an interdisciplinary field of:

- computer science: algorithm, theory, data structure, algorithm implementation, data warehousing, ...
- statistics: algorithm, theoretical insights, modelling, ...
- domain specifics: theoretical and practical insights, special knowledge, ...

Our focus: Mostly implementation and algorithms

Overall Computer Science Approach

Technical Problem	Data Mining problems
↓	
Mathematical Method	Theoretical method to solve problem
↓	
Algorithm	Theoretical algorithm implementing method
↓	
Implementation	Actual implementation

Data Mining: Problems

Our focus: Classification

Given:

- Set of possible classes $\mathcal{Y}$, e.g. $\mathcal{Y} = \{-1, +1\}$
- Set of labelled training examples / data
  $\mathcal{D} = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_N, y_N) \mid (\vec{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$
- A model $f_\theta : \mathcal{X} \to \mathcal{Y}$ with parameter $\theta \in \Theta$

Find: $\widehat{\theta}$, so that $f_{\widehat{\theta}}(\vec{x}) = \widehat{f}(\vec{x})$ that predicts class y for given $\vec{x}$

Data Mining: Problems

Our focus: Classification

Given:

- Set of possible classes $\mathcal{Y}$, e.g. $\mathcal{Y} = \{-1, +1\}$
- Set of labelled training examples / data
  $\mathcal{D} = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_N, y_N) \mid (\vec{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$
- A model $f_\theta : \mathcal{X} \to \mathcal{Y}$ with parameter $\theta \in \Theta$

Find: $\widehat{\theta}$, so that $f_{\widehat{\theta}}(\vec{x}) = \widehat{f}(\vec{x})$ that predicts class y for given $\vec{x}$

Note 1: If $|\mathcal{Y}| = 2$ its called binary classification
Note 2: If $\mathcal{Y} = \mathbb{R}$ its called regression
Our focus: Binary classification: $\mathcal{Y} = \{0, +1\}$ or $\mathcal{Y} = \{-1, +1\}$

Data Mining: Notation

Note: The input space can be (nearly) everything
Our focus: $d-$dimensional vectors: $\vec{x} \in \mathcal{X} \subseteq \mathbb{R}^n$

$\mathcal{D}$	Feature 1	Feature 2	...	Feature d	Label
Example 1	x_{11}	x_{12}	$\dots$	x_{1d}	y_1
Example 2	x_{21}	x_{22}	$\dots$	x_{2d}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Example N	x_{N1}	x_{N1}	$\dots$	x_{Nd}	y_N

Data Mining: Notation

Note: The input space can be (nearly) everything
Our focus: $d-$dimensional vectors: $\vec{x} \in \mathcal{X} \subseteq \mathbb{R}^n$

$\mathcal{D}$	Feature 1	Feature 2	...	Feature d	Label
Example 1	x_{11}	x_{12}	$\dots$	x_{1d}	y_1
Example 2	x_{21}	x_{22}	$\dots$	x_{2d}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Example N	x_{N1}	x_{N1}	$\dots$	x_{Nd}	y_N

Matrix $X \in \mathbb{R}^{d \times N}$

Vector $\vec{y} \in \mathcal{Y}^N$

then: in short $\mathcal{D} = (X, \vec{y})$

Overall Computer Science Approach

Technical Problem	Classification ✓
↓	
Mathematical Method	Theoretical method to solve problem
↓	
Algorithm	Theoretical algorithm implementing method
↓	
Implementation	Actual implementation

What is a good model function?

Observation

We need model function f_θ

What is a good model function?

Observation

We need model function f_θ

Maybe simplest model

$$f(\vec{x}) = \begin{cases} +1 \text{ if } x_i > c \\ -1 \text{ else} \end{cases}$$

What is a good model function?

Observation
We need model function f_θ

Maybe simplest model

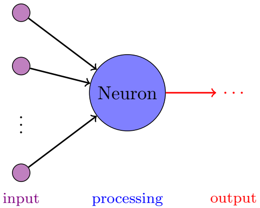
$$f(\vec{x}) = \begin{cases} +1 \text{ if } x_i > c \\ -1 \text{ else} \end{cases}$$

Thus $\theta = (i, c)$
But Which feature is important?
Again simple Just use all

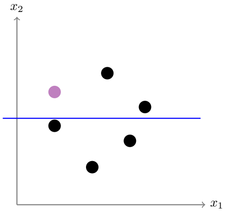
Artificial Neural Networks: Single Neuron

Simple case: Let $\vec{x} \in \mathbb{B}^d$

Biology's view:



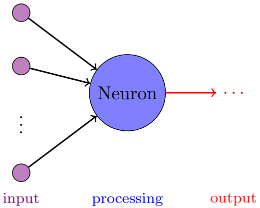
Geometrical view:



Artificial Neural Networks: Single Neuron

Simple case: Let $\vec{x} \in \mathbb{B}^d$

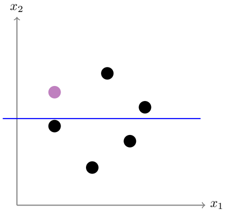
Biology's view:



"Fire" if input signals reach threshold:

$$f(\vec{x}) = \begin{cases} +1 & \text{if } \sum_{i=1}^{d} x_i \geq b \\ 0 & \text{else} \end{cases}$$

Geometrical view:



Predict class depending on side of line (count):

$$f(\vec{x}) = \begin{cases} +1 & \text{if } \sum_{i=1}^{d} x_i \geq b \\ 0 & \text{else} \end{cases}$$

Artificial Neural Networks: Single Neuron

Note: We basically count the number of positive inputs
1943: McCulloch-Pitts Neuron:

- Simple linear model with binary input and output
- Can model boolean OR with $b = 1$
- Can model boolean AND with $b = d$
- Simple extension also allows boolean NOT

Artificial Neural Networks: Single Neuron

Note: We basically count the number of positive inputs

1943: McCulloch-Pitts Neuron:

- Simple linear model with binary input and output
- Can model boolean OR with $b = 1$
- Can model boolean AND with $b = d$
- Simple extension also allows boolean NOT

Thus: A network of McCulloch-Pitts neurons can simulate every boolean function (functional complete)

Artificial Neural Networks: Single Neuron

Note: We basically count the number of positive inputs

1943: McCulloch-Pitts Neuron:

- Simple linear model with binary input and output
- Can model boolean OR with $b = 1$
- Can model boolean AND with $b = d$
- Simple extension also allows boolean NOT

Thus: A network of McCulloch-Pitts neurons can simulate every boolean function (functional complete)

Remark: That does not help with classification, thus

- **Rosenblatt 1958:** Use weights $w_i \in \mathbb{R}$ for every input $x_i \in \mathbb{B}$
- **Minksy-Papert 1959:** Allow real valued inputs $x_i \in \mathbb{R}$

Artificial Neural Networks: Perceptron

A perceptron is a linear classifier $f\colon \mathbb{R}^d \rightarrow \{0,1\}$ with

$$\widehat{f}(\vec{x}) = \begin{cases} +1 & \text{if } \sum_{i=1}^{d} w_i \cdot x_i \geq b \\ 0 & \text{else} \end{cases}$$

Artificial Neural Networks: Perceptron

A perceptron is a linear classifier $f\colon \mathbb{R}^d \rightarrow \{0, 1\}$ with

$$\widehat{f}(\vec{x}) = \begin{cases} +1 & \text{if } \sum_{i=1}^{d} w_i \cdot x_i \geq b \\ 0 & \text{else} \end{cases}$$

Linear function in $d = 2$: $y = mx + \tilde{b}$
Perceptron: $w_1 \cdot x_1 + w_2 \cdot x_2 \geq b \Leftrightarrow x_2 = \frac{b}{w_2} - \frac{w_1}{w_2} x_1$
Obviously: A perceptron is a hyperplane in d dimensions

Artificial Neural Networks: Perceptron

A perceptron is a linear classifier $f\colon \mathbb{R}^d \to \{0,1\}$ with

$$\widehat{f}(\vec{x}) = \begin{cases} +1 & \text{if } \sum_{i=1}^{d} w_i \cdot x_i \geq b \\ 0 & \text{else} \end{cases}$$

Linear function in $d = 2$: $y = mx + \tilde{b}$
Perceptron: $w_1 \cdot x_1 + w_2 \cdot x_2 \geq b \Leftrightarrow x_2 = \frac{b}{w_2} - \frac{w_1}{w_2} x_1$
Obviously: A perceptron is a hyperplane in d dimensions

Note: $\vec{w} = (w_1, \ldots, w_d, b)^T$ are the parameters of a perceptron
Notation: Given $\vec{x}$ we add a 1 to the end of it $\vec{x} = (x_1, \ldots, x_d, 1)^T$

$$\textbf{Then}: \quad \widehat{f}(\vec{x}) = \begin{cases} +1 & \text{if } \vec{x} \cdot \vec{w}^T \geq 0 \\ 0 & \text{else} \end{cases}$$

ANN: Perceptron Learning

Note: A perceptron assumes that the data is linear separable

ANN: Perceptron Learning

Note: A perceptron assumes that the data is linear separable
Big Note: This is an assumption and not necessarily true!

ANN: Perceptron Learning

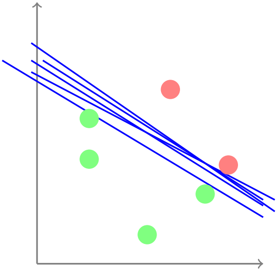
Note: A perceptron assumes that the data is linear separable

Big Note: This is an assumption and not necessarily true!

But: In case of linear separability, there are many “good” $\vec{w}$

ANN: Perceptron Learning

Note: A perceptron assumes that the data is linear separable
Big Note: This is an assumption and not necessarily true!
But: In case of linear separability, there are many “good” $\vec{w}$



Note: We are happy with **one** separative vector $\vec{w}$

Overall Computer Science Approach

Technical Problem	Classification ✓
↓	
Mathematical Method	Perceptron ✓
↓	
Algorithm	Theoretical algorithm implementing method
↓	
Implementation	Actual implementation

ANN: Perceptron Learning

Question: How do we get the weights $\vec{w}$?

ANN: Perceptron Learning

Question: How do we get the weights $\vec{w}$?
Observation: We look at $\vec{x} \cdot \vec{w}^T \geq 0$

- if output was 0 but should have been 1 increment weights
- if output was 1 but should have been 0 decrement weights
- if output was correct, don't change weights

ANN: Perceptron Learning

Question: How do we get the weights $\vec{w}$?
Observation: We look at $\vec{x} \cdot \vec{w}^T \geq 0$

- if output was 0 but should have been 1 increment weights
- if output was 1 but should have been 0 decrement weights
- if output was correct, don't change weights

$\vec{w} = rand(1, \ldots, d+1)$
while ERROR **do**
  for $(\vec{x}_i, y_i) \in \mathcal{D}$ **do**
    $\vec{w} = \vec{w} + \alpha \cdot \vec{x}_i \cdot (y_i - \widehat{f}(\vec{x}_i))$
  end for
end while

ANN: Perceptron Learning

Question: How do we get the weights $\vec{w}$?
Observation: We look at $\vec{x} \cdot \vec{w}^T \geq 0$

- if output was 0 but should have been 1 increment weights
- if output was 1 but should have been 0 decrement weights
- if output was correct, don't change weights

1: $\vec{w} = rand(1, \ldots, d+1)$
2: **while** ERROR **do**
3:     **for** $(\vec{x}_i, y_i) \in \mathcal{D}$ **do**
4:         $\vec{w} = \vec{w} + \alpha \cdot \vec{x}_i \cdot (y_i - \widehat{f}(\vec{x}_i))$
5:     **end for**
6: **end while**

Note: $\alpha \in \mathbb{R}_{>0}$ is a stepsize / learning rate

ANN: Perceptron Learning

Update rule: $\vec{w}_{new} = \vec{w}_{old} + \alpha \cdot \vec{x}_i \cdot (y_i - \widehat{f}_{old}(\vec{x}_i))$

ANN: Perceptron Learning

Update rule: $\vec{w}_{new} = \vec{w}_{old} + \alpha \cdot \vec{x}_i \cdot (y_i - \widehat{f}_{old}(\vec{x}_i))$

Wrong classification:

- **Case 1:** $y_i - \widehat{f}_{old}(\vec{x}_i) = 1 \Rightarrow y_i = 1, \widehat{f}_{old}(\vec{x}_i) = 0$

ANN: Perceptron Learning

Update rule: $\vec{w}_{new} = \vec{w}_{old} + \alpha \cdot \vec{x}_i \cdot (y_i - \widehat{f}_{old}(\vec{x}_i))$

Wrong classification:

- **Case 1:** $y_i - \widehat{f}_{old}(\vec{x}_i) = 1 \Rightarrow y_i = 1, \widehat{f}_{old}(\vec{x}_i) = 0$

$$\widehat{f}_{new}(\vec{x}_i) \quad = \quad \vec{x}_i \cdot (\vec{w}_{new})^T = \vec{x}_i \cdot (\vec{w}_{old} + \alpha \cdot 1 \cdot \vec{x}_i)^T$$

ANN: Perceptron Learning

Update rule: $\vec{w}_{new} = \vec{w}_{old} + \alpha \cdot \vec{x}_i \cdot (y_i - \widehat{f}_{old}(\vec{x}_i))$

Wrong classification:

- **Case 1:** $y_i - \widehat{f}_{old}(\vec{x}_i) = 1 \Rightarrow y_i = 1, \widehat{f}_{old}(\vec{x}_i) = 0$

$$\begin{aligned}
\widehat{f}_{new}(\vec{x}_i) &= \vec{x}_i \cdot (\vec{w}_{new})^T = \vec{x}_i \cdot (\vec{w}_{old} + \alpha \cdot 1 \cdot \vec{x}_i)^T \\
&= \vec{x}_i \cdot \vec{w}_{old}^T + \alpha \cdot \vec{x}_i \cdot \vec{x}_i^T = \vec{x}_i \cdot \vec{w}_{old}^T + \alpha \cdot ||\vec{x}_i||^2
\end{aligned}$$

ANN: Perceptron Learning

Update rule: $\vec{w}_{new} = \vec{w}_{old} + \alpha \cdot \vec{x}_i \cdot (y_i - \widehat{f}_{old}(\vec{x}_i))$

Wrong classification:

- **Case 1:** $y_i - \widehat{f}_{old}(\vec{x}_i) = 1 \Rightarrow y_i = 1, \widehat{f}_{old}(\vec{x}_i) = 0$

$$
\begin{aligned}
\widehat{f}_{new}(\vec{x}_i) &= \vec{x}_i \cdot (\vec{w}_{new})^T = \vec{x}_i \cdot (\vec{w}_{old} + \alpha \cdot 1 \cdot \vec{x}_i)^T \\
&= \vec{x}_i \cdot \vec{w}_{old}^T + \alpha \cdot \vec{x}_i \cdot \vec{x}_i^T = \vec{x}_i \cdot \vec{w}_{old}^T + \alpha \cdot ||\vec{x}_i||^2
\end{aligned}
$$

$\rightarrow \vec{w}$ is incremented and classification is moved towards 1 ✓

ANN: Perceptron Learning

Update rule: $\vec{w}_{new} = \vec{w}_{old} + \alpha \cdot \vec{x}_i \cdot (y_i - \widehat{f}_{old}(\vec{x}_i))$

Wrong classification:

- **Case 1:** $y_i - \widehat{f}_{old}(\vec{x}_i) = 1 \Rightarrow y_i = 1, \widehat{f}_{old}(\vec{x}_i) = 0$

$$\begin{aligned} \widehat{f}_{new}(\vec{x}_i) &= \vec{x}_i \cdot (\vec{w}_{new})^T = \vec{x}_i \cdot (\vec{w}_{old} + \alpha \cdot 1 \cdot \vec{x}_i)^T \\ &= \vec{x}_i \cdot \vec{w}_{old}^T + \alpha \cdot \vec{x}_i \cdot \vec{x}_i^T = \vec{x}_i \cdot \vec{w}_{old}^T + \alpha \cdot ||\vec{x}_i||^2 \end{aligned}$$

  $\rightarrow \vec{w}$ is incremented and classification is moved towards 1 ✓

- **Case 2:** $y_i - \widehat{f}_{old}(\vec{x}_i) = -1 \Rightarrow y_i = 0, \widehat{f}_{old}(\vec{x}_i) = 1$

ANN: Perceptron Learning

Update rule: $\vec{w}_{new} = \vec{w}_{old} + \alpha \cdot \vec{x}_i \cdot (y_i - \widehat{f}_{old}(\vec{x}_i))$

Wrong classification:

- **Case 1:** $y_i - \widehat{f}_{old}(\vec{x}_i) = 1 \Rightarrow y_i = 1, \widehat{f}_{old}(\vec{x}_i) = 0$

$$\begin{aligned}\widehat{f}_{new}(\vec{x}_i) &= \vec{x}_i \cdot (\vec{w}_{new})^T = \vec{x}_i \cdot (\vec{w}_{old} + \alpha \cdot 1 \cdot \vec{x}_i)^T \\ &= \vec{x}_i \cdot \vec{w}_{old}^T + \alpha \cdot \vec{x}_i \cdot \vec{x}_i^T = \vec{x}_i \cdot \vec{w}_{old}^T + \alpha \cdot ||\vec{x}_i||^2\end{aligned}$$

  $\rightarrow \vec{w}$ is incremented and classification is moved towards 1 ✓

- **Case 2:** $y_i - \widehat{f}_{old}(\vec{x}_i) = -1 \Rightarrow y_i = 0, \widehat{f}_{old}(\vec{x}_i) = 1$

$$\widehat{f}_{new}(\vec{x}_i) = \vec{x}_i \cdot (\vec{w}_{new})^T = \vec{x}_i \cdot (\vec{w}_{old} - \alpha \cdot 1 \cdot \vec{x}_i)^T$$

ANN: Perceptron Learning

Update rule: $\vec{w}_{new} = \vec{w}_{old} + \alpha \cdot \vec{x}_i \cdot (y_i - \widehat{f}_{old}(\vec{x}_i))$

Wrong classification:

- **Case 1:** $y_i - \widehat{f}_{old}(\vec{x}_i) = 1 \Rightarrow y_i = 1, \widehat{f}_{old}(\vec{x}_i) = 0$

$$\begin{aligned}\widehat{f}_{new}(\vec{x}_i) &= \vec{x}_i \cdot (\vec{w}_{new})^T = \vec{x}_i \cdot (\vec{w}_{old} + \alpha \cdot 1 \cdot \vec{x}_i)^T \\ &= \vec{x}_i \cdot \vec{w}_{old}^T + \alpha \cdot \vec{x}_i \cdot \vec{x}_i^T = \vec{x}_i \cdot \vec{w}_{old}^T + \alpha \cdot ||\vec{x}_i||^2\end{aligned}$$

  $\rightarrow \vec{w}$ is incremented and classification is moved towards 1 ✓

- **Case 2:** $y_i - \widehat{f}_{old}(\vec{x}_i) = -1 \Rightarrow y_i = 0, \widehat{f}_{old}(\vec{x}_i) = 1$

$$\begin{aligned}\widehat{f}_{new}(\vec{x}_i) &= \vec{x}_i \cdot (\vec{w}_{new})^T = \vec{x}_i \cdot (\vec{w}_{old} - \alpha \cdot 1 \cdot \vec{x}_i)^T \\ &= \vec{x}_i \cdot \vec{w}_{old}^T - \alpha \cdot \vec{x}_i \cdot \vec{x}_i^T = \vec{x}_i \cdot \vec{w}_{old}^T - \alpha \cdot ||\vec{x}_i||^2\end{aligned}$$

ANN: Perceptron Learning

Update rule: $\vec{w}_{new} = \vec{w}_{old} + \alpha \cdot \vec{x}_i \cdot (y_i - \widehat{f}_{old}(\vec{x}_i))$

Wrong classification:

- **Case 1:** $y_i - \widehat{f}_{old}(\vec{x}_i) = 1 \Rightarrow y_i = 1, \widehat{f}_{old}(\vec{x}_i) = 0$

$$\begin{aligned}\widehat{f}_{new}(\vec{x}_i) &= \vec{x}_i \cdot (\vec{w}_{new})^T = \vec{x}_i \cdot (\vec{w}_{old} + \alpha \cdot 1 \cdot \vec{x}_i)^T \\ &= \vec{x}_i \cdot \vec{w}_{old}^T + \alpha \cdot \vec{x}_i \cdot \vec{x}_i^T = \vec{x}_i \cdot \vec{w}_{old}^T + \alpha \cdot ||\vec{x}_i||^2\end{aligned}$$

  $\rightarrow \vec{w}$ is incremented and classification is moved towards 1 ✓

- **Case 2:** $y_i - \widehat{f}_{old}(\vec{x}_i) = -1 \Rightarrow y_i = 0, \widehat{f}_{old}(\vec{x}_i) = 1$

$$\begin{aligned}\widehat{f}_{new}(\vec{x}_i) &= \vec{x}_i \cdot (\vec{w}_{new})^T = \vec{x}_i \cdot (\vec{w}_{old} - \alpha \cdot 1 \cdot \vec{x}_i)^T \\ &= \vec{x}_i \cdot \vec{w}_{old}^T - \alpha \cdot \vec{x}_i \cdot \vec{x}_i^T = \vec{x}_i \cdot \vec{w}_{old}^T - \alpha \cdot ||\vec{x}_i||^2\end{aligned}$$

  $\rightarrow \vec{w}$ is decremented and classification is moved towards 0 ✓

ANN: Perceptron Learning

Update rule: $\vec{w}_{new} = \vec{w}_{old} + \alpha \cdot \vec{x}_i \cdot (y_i - \widehat{f}_{old}(\vec{x}_i))$

ANN: Perceptron Learning

Update rule: $\vec{w}_{new} = \vec{w}_{old} + \alpha \cdot \vec{x}_i \cdot (y_i - \widehat{f}_{old}(\vec{x}_i))$
Correct classification: $y_i - \widehat{f}(\vec{x}_i) = 0$

- $\vec{w}_{new} = \vec{w}_{old}$, thus $\vec{w}$ is unchanged ✓

ANN: Perceptron Learning

Update rule: $\vec{w}_{new} = \vec{w}_{old} + \alpha \cdot \vec{x}_i \cdot (y_i - \widehat{f}_{old}(\vec{x}_i))$
Correct classification: $y_i - \widehat{f}(\vec{x}_i) = 0$

- $\vec{w}_{new} = \vec{w}_{old}$, thus $\vec{w}$ is unchanged ✓

Rosenblatt 1958 showed:

- Algorithms converges if $\mathcal{D}$ is linear separable
- Algorithm may have exponential runtime

ANN: Perceptron Learning

Update rule: $\vec{w}_{new} = \vec{w}_{old} + \alpha \cdot \vec{x}_i \cdot (y_i - \widehat{f}_{old}(\vec{x}_i))$
Correct classification: $y_i - \widehat{f}(\vec{x}_i) = 0$

- $\vec{w}_{new} = \vec{w}_{old}$, thus $\vec{w}$ is unchanged ✓

Rosenblatt 1958 showed:

- Algorithms converges if $\mathcal{D}$ is linear separable
- Algorithm may have exponential runtime

Variation: Batch processing - Update $\vec{w}$ after testing all examples

$$\vec{w}_{new} = \vec{w}_{old} + \alpha \sum_{(\vec{x}_i, y_i) \in \mathcal{D}_{wrong}} \vec{x}_i \cdot (y_i - \widehat{f}_{old}(\vec{x}_i))$$

Usually: Faster convergence, but more memory needed

Overall Computer Science Approach

Technical Problem	Classification ✓
↓	
Mathematical Method	Linear classifier ✓
↓	
Algorithm	Perceptron learning ✓
↓	
Implementation	Actual implementation

Data Mining: Implementation of Perceptron Learning

Obviously: Implementation also influences the runtime!

Data Mining: Implementation of Perceptron Learning

Obviously: Implementation also influences the runtime!

Fact: We need to take the underlying system into account

- **System:** CPU, GPU, FPGA, ...
- **Hardware:** Word length, cache sizes, vectorization, ...
- **Software:** Paging in OS, (Multi-) Threading, Swapping, ...
- **Language:** `C` vs. `Java` vs. `Haskell` ...

Data Mining: Implementation of Perceptron Learning

Obviously: Implementation also influences the runtime!

Fact: We need to take the underlying system into account

- **System:** CPU, GPU, FPGA, ...
- **Hardware:** Word length, cache sizes, vectorization, ...
- **Software:** Paging in OS, (Multi-) Threading, Swapping, ...
- **Language:** `C` vs. `Java` vs. `Haskell` ...

Usually: Use language and system we know
But: Some systems / hardware is better at certain tasks
$\rightarrow$ e.g. graphics cards are built to do matrix-vector multiplication

Data Mining: Implementation of Perceptron Learning

Obviously: Implementation also influences the runtime!

Fact: We need to take the underlying system into account

- **System:** CPU, GPU, FPGA, ...
- **Hardware:** Word length, cache sizes, vectorization, ...
- **Software:** Paging in OS, (Multi-) Threading, Swapping, ...
- **Language:** `C` vs. `Java` vs. `Haskell` ...

Usually: Use language and system we know
But: Some systems / hardware is better at certain tasks
→ e.g. graphics cards are built to do matrix-vector multiplication

Thus: Choose method and algorithm depending on system
Our focus: Mostly methods and algorithms, later implementation

Overall Computer Science Approach

Technical Problem	Classification ✓
↓	
Mathematical Method	Perceptron ✓
↓	
Algorithm	Simple learning rule ✓
↓	
Implementation	System and language ✓

Data Mining: Measure Model quality

Fact 1: Prediction quality also depends on the algorithm, the implementation and the data
$\rightarrow$ Integer operations are fast, but less accurate than floating point

Data Mining: Measure Model quality

Fact 1: Prediction quality also depends on the algorithm, the implementation and the data
$\rightarrow$ Integer operations are fast, but less accurate than floating point

Fact 2: There are many different models, even more algorithms and even more implementations
$\rightarrow$ Learning Rule, Gradient Descent, Evolutionary Optimization ...

Data Mining: Measure Model quality

Fact 1: Prediction quality also depends on the algorithm, the implementation and the data
$\rightarrow$ Integer operations are fast, but less accurate than floating point

Fact 2: There are many different models, even more algorithms and even more implementations
$\rightarrow$ Learning Rule, Gradient Descent, Evolutionary Optimization ...

Bottom line: Comparing specific methods is difficult
Thus: Compare performance of **computed** model

Data Mining: Measure Model quality

Fact 1: Prediction quality also depends on the algorithm, the implementation and the data
→ Integer operations are fast, but less accurate than floating point

Fact 2: There are many different models, even more algorithms and even more implementations
→ Learning Rule, Gradient Descent, Evolutionary Optimization ...

Bottom line: Comparing specific methods is difficult
Thus: Compare performance of **computed** model

Important: There is no free lunch (**Wolpert, 1996**)
→ Some methods work better on some problems, but no method works well on all problems

Data Mining: Measure Model quality (2)

Question: So, what is model quality?

Data Mining: Measure Model quality (2)

Question: So, what is model quality?

1. how well explains the model training data?
2. can we give any guarantees for new predictions?
3. how well generalises the model to new and unseen data?

So far Linear model assumption
No guarantees at all, especially if linear assumption does not hold

Data Mining: Measure Model quality (3)

Fact: In binary classification we have two choices: predict 0 or 1
$\rightarrow$ 2 possible wrong predictions and 2 possible correct predictions

Data Mining: Measure Model quality (3)

Fact: In binary classification we have two choices: predict 0 or 1
$\rightarrow$ 2 possible wrong predictions and 2 possible correct predictions
Visualization: Confusion matrix

	Predicted value	
True value	True positive (TP)	False negative (FN)
	False positive (FP)	True negative (TN)

Data Mining: Measure Model quality (3)

Fact: In binary classification we have two choices: predict 0 or 1
$\rightarrow$ 2 possible wrong predictions and 2 possible correct predictions
Visualization: Confusion matrix

	Predicted value	
True value	True positive (TP)	False negative (FN)
	False positive (FP)	True negative (TN)

Accuracy: $Acc = \frac{TP+TN}{N}$

Big Remark: The accuracy only tells us something about the data $\mathcal{D}$ we know! There are no guarantees for new data

Data Mining: Measure Model quality (4)

Obviously: The best model has $Acc = 1$, the worst has $Acc = 0$
Observation: If we store all the data for look-up, then $Acc = 1$

Data Mining: Measure Model quality (4)

Obviously: The best model has $Acc = 1$, the worst has $Acc = 0$
Observation: If we store all the data for look-up, then $Acc = 1$

Question: Is that what we want?
Clear: This is just memorizing the training data, no real learning!
Question: How well deals our model with new, yet unseen data?

Data Mining: Measure Model quality (4)

Obviously: The best model has $Acc = 1$, the worst has $Acc = 0$
Observation: If we store all the data for look-up, then $Acc = 1$

Question: Is that what we want?
Clear: This is just memorizing the training data, no real learning!
Question: How well deals our model with new, yet unseen data?

Idea: Split data into training $\mathcal{D}_{Train}$ and test data $\mathcal{D}_{Test}$
Then: $\mathcal{D}_{Test}$ is new to the model $f_{\widehat{\theta}}$
Question: How to split $\mathcal{D}$?

Data Mining: Measure Model quality (5)

1) Test/Train: Split $\mathcal{D}$ by size, e.g. 80% training and 20% test data
$\rightarrow$ Fast and easy to compute, but sensitive for “bad” splits.
$\rightarrow$ Model quality might be over- or under-estimated

Data Mining: Measure Model quality (5)

1) Test/Train: Split $\mathcal{D}$ by size, e.g. 80% training and 20% test data
$\rightarrow$ Fast and easy to compute, but sensitive for "bad" splits.
$\rightarrow$ Model quality might be over- or under-estimated

2) Leave-One-Out: Use every example once for testing and train model on the remaining data. Average results.
$\rightarrow$ N models are computed, but insensitive for "bad" splits.
$\rightarrow$ Usually impractical

Data Mining: Measure Model quality (5)

1) Test/Train: Split $\mathcal{D}$ by size, e.g. 80% training and 20% test data
$\rightarrow$ Fast and easy to compute, but sensitive for "bad" splits.
$\rightarrow$ Model quality might be over- or under-estimated

2) Leave-One-Out: Use every example once for testing and train model on the remaining data. Average results.
$\rightarrow$ N models are computed, but insensitive for "bad" splits.
$\rightarrow$ Usually impractical

3) K-fold cross validation: Split data into k buckets. Use every bucket once for testing / train model on the rest. Average results.
$\rightarrow$ Insensitive for "bad" splits and practical. Usually $k = 10$.

Summary

Important concepts:

- **Classification** is one data mining task
- **Training data** is used to define and solve the task
- **A Method** is a general approach / idea to solve a task
- **A algorithm** is a way to realise a method
- **A model** forms the extracted knowledge from data
- **Accuracy** measures the model quality given the data

Summary

Important concepts:

- **Classification** is one data mining task
- **Training data** is used to define and solve the task
- **A Method** is a general approach / idea to solve a task
- **A algorithm** is a way to realise a method
- **A model** forms the extracted knowledge from data
- **Accuracy** measures the model quality given the data

Note: Runtime and model quality depend on method, algorithm and implementation

Some administration stuff

Requirements to pass this course

- Plan an approach to solve kaggle competition including
  - Data pre-processing
  - Implementation of Neural Network learning
  - Incorperate FPGA design
- Give a small presentation / review about your approach

Some administration stuff

Requirements to pass this course

- Plan an approach to solve kaggle competition including
  - Data pre-processing
  - Implementation of Neural Network learning
  - Incorperate FPGA design
- Give a small presentation / review about your approach

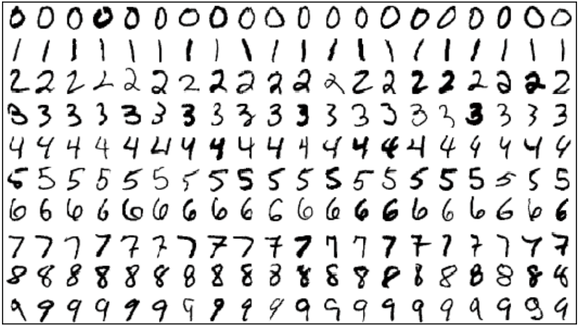
Thus: After the lecture phase you are free to do what you want until the end of the semester → you work in self-organizing groups

Question: When will we meet again for lectures?

Homwork Data

For development Use smaller data set

- 32×32 pixel grayscaled images of numbers $0 - 9$ $(10$ labels$)$
- already pre-processed in CSV format
- test/train split plus a smaller sample for development



Homework

Homework I give simple homeworks to get you started more easily
But I will not check the homework, your choice to do it.

Homework

Homework I give simple homeworks to get you started more easily
But I will not check the homework, your choice to do it.

Homework until next meeting

- Implement a simple CSV-Reader
  - First column contains the label $(0 - 9)$
  - Remaining 784 columns contain grayscale value $(0 - 255)$
- Implement perceptron learning algorithm for two numbers
- Implement accuracy computation for Test/Train split

Note 1: We will later use `C`, so please use `C` or a `C`-like language
Note 2: Use the smaller split for development and the complete data set for testing $\rightarrow$ What's your accuracy?