# Lecture 1: Introduction

Jakub M. Tomczak

Deep learning

VU VRIJE UNIVERSITEIT AMSTERDAM

# COURSE ORGANIZATION

# DO NOT HESITATE TO CONTACT US!

If you have any question, please contact us:

TAs:

- Dimitrios Alivanistos
- Daniel Daza
- Emile van Krieken
- Anna Kuzina
- Stefan Schouten
- Shuai Wang
- Roderick van der Weerdt
- Taraneh Younesian

Lecturers:

Peter Bloem

Michael Cochez

Jakub Tomczak

# LECTURES AND PRACTICAL SESSIONS

During this course we will have:

- 14 lectures (incl. 2 invited lectures)
- 7 days with practical sessions (1 day, 5 time slots)
- 1 meeting for the final exam

Content of the course:

1. From logistic regression to fully-connected networks
2. Convolutional nets, recurrent neural nets
3. Generative modeling: GANs, VAEs, autoregressive models, flows
4. Graph convolutions, self-attention, transformers
5. Reinforcement Learning

# LECTURES AND PRACTICAL SESSIONS

Assignments:

1. MLP (Nov 14)
2. Autograd/backpropagation (Nov 28)
3. One of the two (Dec 8):
   a. CNNs
   b. RNNs
4. One of the three (Dec 19):
   a. VAEs & GANs
   b. Graph convolutions
   c. Reinforcement learning

Assignments:

1. Use Python 3 ONLY!
2. Assignments 1 and 2 are implemented INDIVIDUALLY.
3. Assignments 3 and 4 are implemented IN GROUPS OF 3.
4. All methods must be implemented by you unless it's specified otherwise.
5. In the assignments we will use mainly Numpy and PyTorch.
6. THERE IS NO RESIT FOR ASSIGNMENTS.
7. Late submissions: First contact Jakub. Second, -2pts for each day after the deadline.

# GRADING

**Assignments**

- Max. 40 pts (10 pts per assignment)
- Partial grade from the assignments: round(achieved points / 4)

**Exam**

- Max. 40 pts (40 questions)
- Partial grade: round(achieved points / 4)

**Final grade**

- At least 5.5 from the assignments and 5.5 from the exam.
- Final grade:
  round(0.5 * partial grade from assignments + 0.5 * partial grade from the exam)

# HISTORICAL PERSPECTIVE

**1940-1950**
Cellular automata
(von Neumann-Ulam)

**1948**
Information theory
(Shannon)

**1950**
Turing test
(Turing)

**1956**
Term “AI”
(McCarthy)

**1958**
Perceptron
(Rosenblatt)

**1964**
ELIZA
(Weizenbaum)

**1960s and 1970s**
Evolutionary computation
(Fogel, Holland,
Rechenberg, Schwefel)

**1970s**
Expert systems
(e.g., Quinlan, Michalski)

**1982**
ConvNets
(Fukushima)

**1985**
Bayesian Nets
(Pearl)

**1986**
Backpropagation
(Rumelhart et al.)

**1995**
SVMs
(Cortes&Vapnik)

**1995**
Deep Blue
Kasparov

**1997**
LSTM
(Hochreiter&Schmidhüber)

**2000-…**
Deep learning era

# WHY AI IS SUCCESSFUL?

Accessible hardware

Powerful hardware

Intuitive programming languages

Specialized packages

**Knowledge representation**

How to represent & process data?

**Knowledge acquisition (learning objective & algorithms)**

How to extract knowledge?

**Learning problems**

What kind of problems can we formulate?

# LEARNING AS OPTIMIZATION

For given **data** $\mathscr{D}$, find the **best data representation** from a given **class of representations** that minimizes given **learning objective (loss)**.

$$\min_{x \in \mathbb{X}} f(x; \mathscr{D})$$

$$\text{s.t. } x \in \mathbb{Y} \subseteq \mathbb{X}$$

For given **data** $\mathscr{D}$, find the **best data representation** from a given **class of representations** that minimizes given **learning objective (loss)**.

$$\min_{x \in \mathbb{X}} \; f(x; \mathscr{D})$$

$$\text{s.t. } x \in \mathbb{Y} \subseteq \mathbb{X}$$

**Optimization algorithm = learning algorithm.**

# LEARNING TASKS

## Supervised Learning

- We distinguish **inputs** and **outputs**.
- We are interested in **prediction**.
- We minimize a **prediction error.**

# LEARNING TASKS

## Supervised Learning

- We distinguish **inputs** and **outputs**.
- We are interested in **prediction**.
- We minimize a **prediction error.**

## Unsupervised learning

- **No** distinction among variables.
- We look for a **data structure**.
- We minimize a **reconstruction error**, **compression rate**, …

# LEARNING TASKS

### Supervised Learning

- We distinguish **inputs** and **outputs**.
- We are interested in **prediction**.
- We minimize a **prediction error.**

### Unsupervised learning

- **No** distinction among variables.
- We look for a **data structure**.
- We minimize a **reconstruction error**, **compression rate**, …

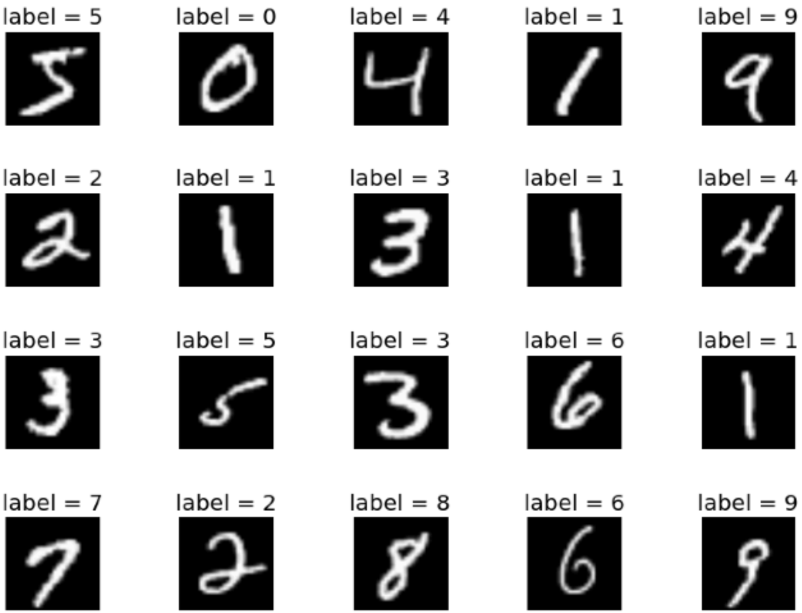
### Reinforcement learning

- An **agent** interacts with an **environment**.
- We want to learn a **policy**.
- Each **action** is **rewarded.**
- We maximize a **total reward**.

## Supervised Learning

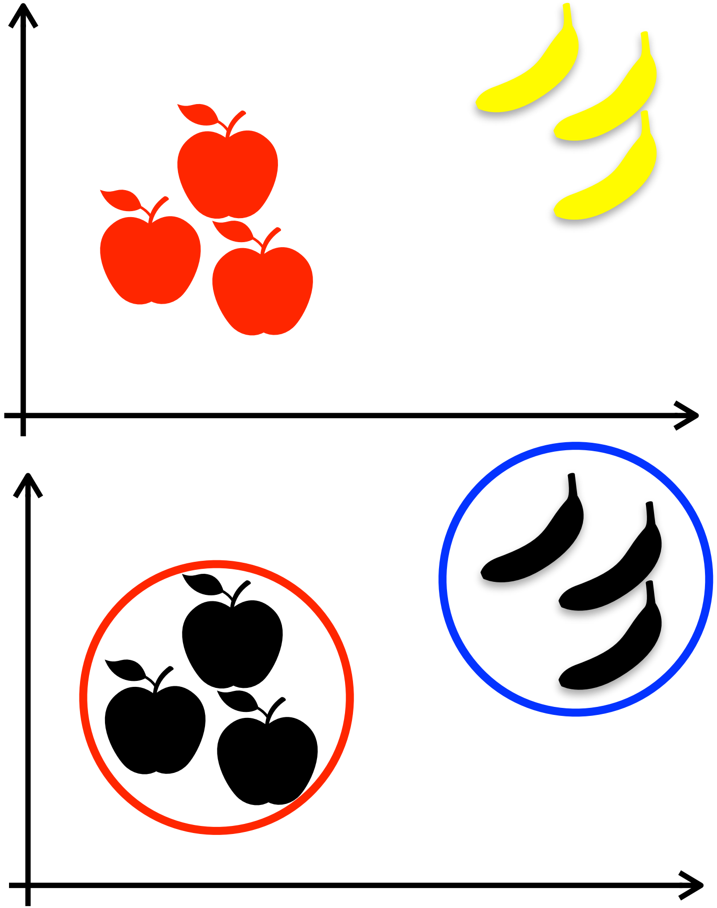
- We distinguish **inputs** and **outputs**.
- We are interested in **prediction**.
- We minimize a **prediction error.**

Classification



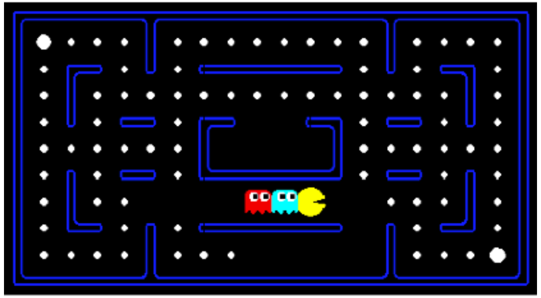
## Unsupervised learning

- **No** distinction among variables.
- We look for a **data structure**.
- We minimize a **reconstruction error**, **compression rate**, …



## Reinforcement learning

- An **agent** interacts with an **environment**.
- We want to learn a **policy**.
- Each **action** is **rewarded.**
- We maximize a **total reward**.



# APPLICATION AREAS

Computer Vision

Information Retrieval

Speech Recognition

Natural Language Processing

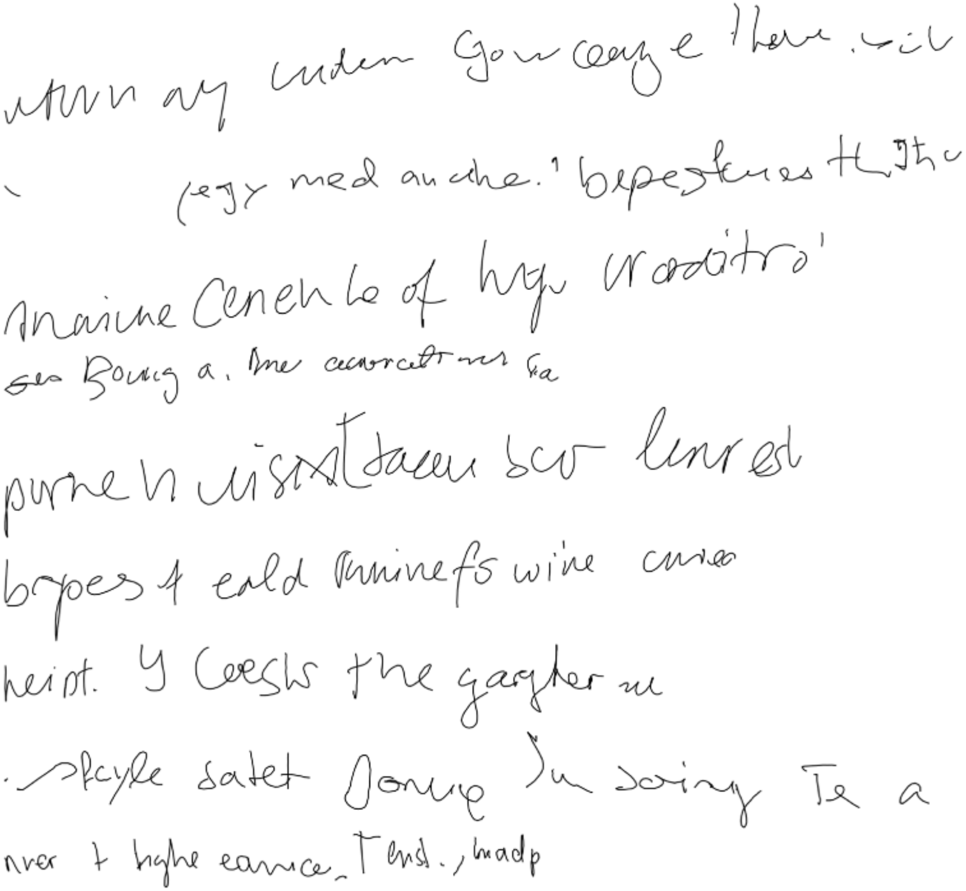
Recommendation Systems

Drug Discovery

Robotics

…

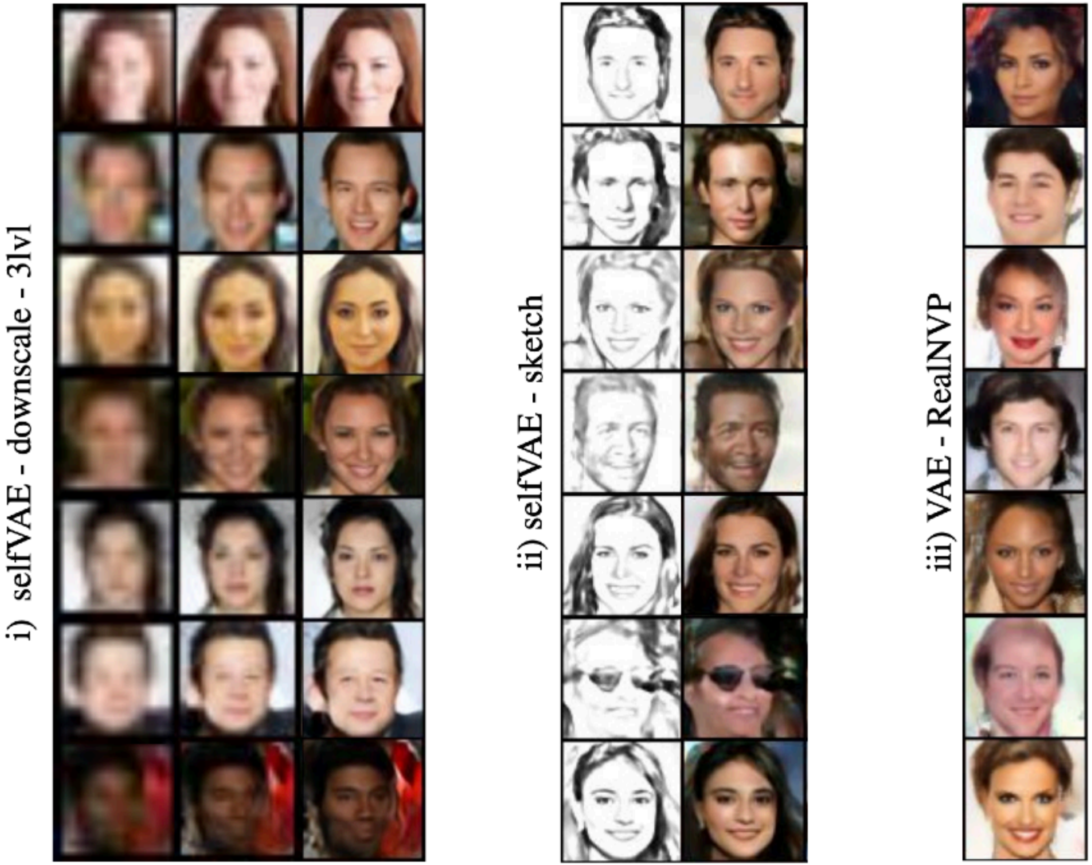
# EXAMPLES: HANDWRITING GENERATOR





A. Graves, “Generating Sequences With Recurrent Neural Networks“

# EXAMPLES: IMAGE GENERATION






Gatopoulos & Tomczak, “Self-supervised Variational Auto-Encoders“

# EXAMPLES: GENERATING IMAGE DESCRIPTIONS

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

"girl in pink dress is jumping in air."

"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

"man in blue wetsuit is surfing on wave."



Karpathy & Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions"

# THREE CORE COMPONENTS OF DEEP LEARNING

- The INTERNET
- Unlabeled data
- ImageNet database
14,197,122 images

IMAGENET

- Parallel computing
- GPU, FPGA
- Deep learning frameworks

theano Caffe

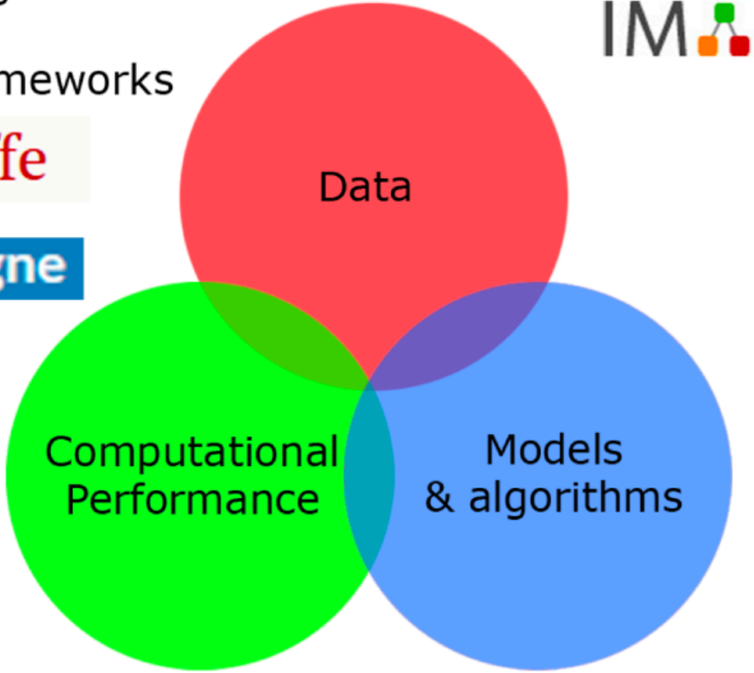
Lasagne

TensorFlow

torch

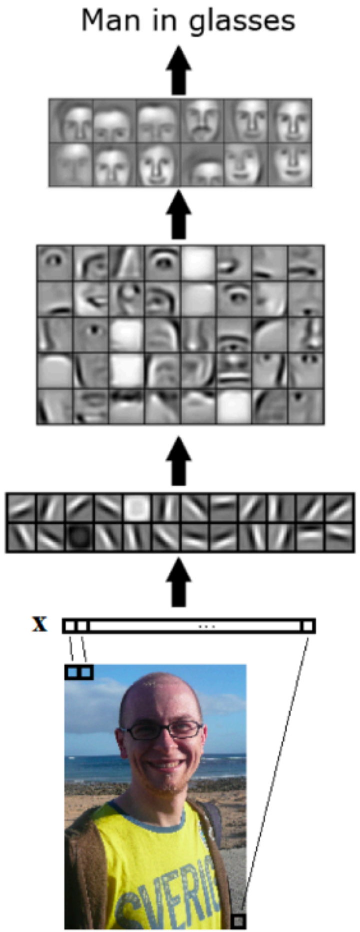
PYTORCH

cuDNN



# AUTOMATIC FEATURE EXTRACTION / REPRESENTATION LEARNING

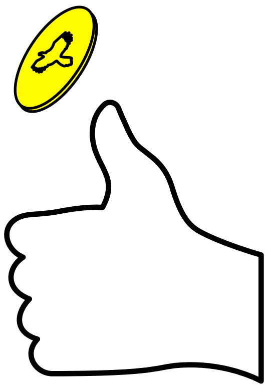
- A **representation** = a set of (abstract) features.
- Deep learning automatically extract feature.
- A hidden layer = an abstraction level.
- Good-quality features should be:
  - **Informative** (e.g., discriminative);
  - **Robust** to small perturbations;
  - **Invariant**/**Equivariant** to transformations.
- **Representation Learning** = optimization algo.



## PROBABILISTIC LEARNING

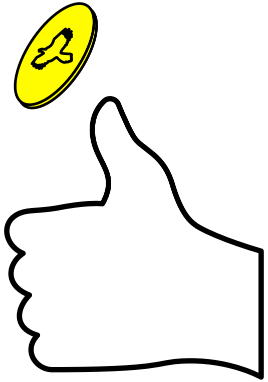
# TOSS A COIN… 🎶

- $c \in \{0,1\}$ - a **random variable** (a result of tossing a coin)



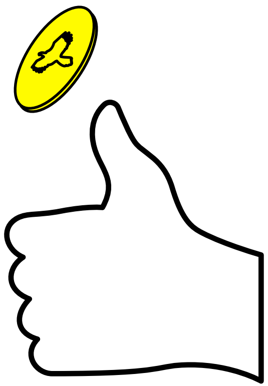
# TOSS A COIN… 🎶

- $c \in \{0,1\}$ - a **random variable** (a result of tossing a coin)
- $x = p(c = 1)$ - **probability** of observing *head*
- $1 - x = p(c = 0)$ - probability of observing *tail*
- $p(c\,|\,x) = x^c\,(1 - x)^{1-c}$ - **Bernoulli distribution**



# TOSS A COIN… 🎶

- $c \in \{0,1\}$ - a **random variable** (a result of tossing a coin)
- $x = p(c = 1)$ - **probability** of observing *head*
- $1 - x = p(c = 0)$ - probability of observing *tail*
- $p(c|x) = x^c\,(1-x)^{1-c}$ - **Bernoulli distribution**



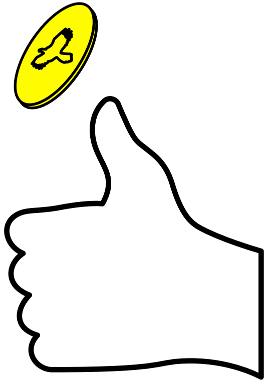
**Quick check:**

$$p(c = 0|x) = x^0\,(1-x)^{1-0} = 1 - x$$

$$p(c = 1|x) = x^1\,(1-x)^{1-1} = x$$

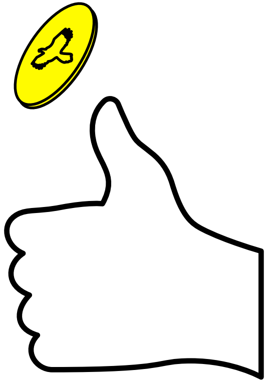
# TOSS A COIN... 🎶

- $c \in \{0,1\}$ - a **random variable** (a result of tossing a coin)
- $x = p(c = 1)$ - **probability** of observing *head*
- $1 - x = p(c = 0)$ - probability of observing *tail*
- $p(c \,|\, x) = x^c \,(1 - x)^{1-c}$ - **Bernoulli distribution**
- $\mathscr{D} = \{c_1, c_2, \ldots, c_N\}$ - *iid* observations (**data**)



# TOSS A COIN... 🎶

- $c \in \{0,1\}$ - a **random variable** (a result of tossing a coin)
- $x = p(c = 1)$ - **probability** of observing *head*
- $1 - x = p(c = 0)$ - probability of observing *tail*
- $p(c\,|\,x) = x^{c}\,(1 - x)^{1-c}$ - **Bernoulli distribution**
- $\mathscr{D} = \{c_1, c_2, \ldots, c_N\}$ - *iid* observations (**data**)

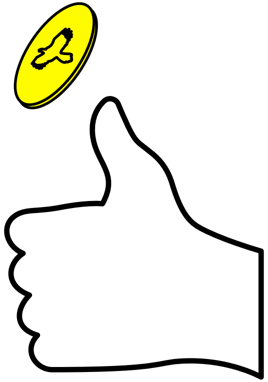


**EXAMPLE:**

$\mathscr{D} = \{0,0,1,1,0,1,1\}$

- $c \in \{0,1\}$ - a **random variable** (a result of tossing a coin)
- $x = p(c = 1)$ - **probability** of observing *head*
- $1 - x = p(c = 0)$ - probability of observing *tail*
- $p(c\,|\,x) = x^c\,(1-x)^{1-c}$ - **Bernoulli distribution**
- $\mathscr{D} = \{c_1, c_2, \ldots, c_N\}$ - *iid* observations (**data**)
- The **likelihood function**:

$$p(\mathscr{D}\,|\,x) = \prod_{n=1}^{N} p(c_n\,|\,x)$$



**The optimization problem**:

**The optimization problem**:

Find such $x \in [0,1]$ that minimizes the **negative log-likelihood function**:

$$\min_{x \in [0,1]} \; -\log p(\mathscr{D} \mid x)$$

**The optimization problem**:

Find such $x \in [0,1]$ that minimizes the **negative log-likelihood function**:

$$\min_{x\in[0,1]} \; -\log p(\mathscr{D} \mid x)$$

**Remarks**:

**The optimization problem**:

Find such $x \in [0,1]$ that minimizes the **negative log-likelihood function**:

$$\min_{x \in [0,1]} -\log p(\mathscr{D} \mid x)$$

**Remarks**:

1) Why **negative**? Because: $\max f(x) = \min \{-f(x)\}$.

**The optimization problem**:

Find such $x \in [0,1]$ that minimizes the **negative log-likelihood function**:

$$\min_{x\in[0,1]} \; -\log p(\mathscr{D} \mid x)$$

**Remarks**:

1) Why **negative**? Because: $\max f(x) = \min \{-f(x)\}$.

2) Why **logarithm**? Because: $\log \prod = \sum \log$ and optimum is the same.

# TOSS A COIN... 🎶

$$\log p(\mathscr{D} \mid x) = \log \prod_{n=1}^{N} p(c_n \mid x)$$

the log-likelihood

# TOSS A COIN… 🎶

$$\log p(\mathscr{D} \mid x) = \log \prod_{n=1}^{N} p(c_n \mid x)$$

$$= \sum_{n=1}^{N} \log p(c_n \mid x)$$

the log-likelihood

$$\log \prod = \sum \log$$

# TOSS A COIN… 🎶

$$\log p(\mathscr{D} \mid x) = \log \prod_{n=1}^{N} p(c_n \mid x)$$

the log-likelihood

$$= \sum_{n=1}^{N} \log p(c_n \mid x)$$

$$\log \prod = \sum \log$$

$$= \sum_{n=1}^{N} \log x^{c_n} \, (1 - x)^{1-c_n}$$

Bernoulli distribution

# TOSS A COIN… 🎶

$$\log p(\mathscr{D} \mid x) = \log \prod_{n=1}^{N} p(c_n \mid x)$$

the log-likelihood

$$= \sum_{n=1}^{N} \log p(c_n \mid x)$$

$$\log \prod = \sum \log$$

$$= \sum_{n=1}^{N} \log x^{c_n} (1-x)^{1-c_n}$$

Bernoulli distribution

$$= \sum_{n=1}^{N} \Big( c_n \log x + (1-c_n) \log(1-x) \Big)$$

$$\log a^b = b \log a$$

$$\log ab = \log a + \log b$$

# TOSS A COIN... 🎶

(Finding $x^\star$) Calculate derivative wrt $x$ and set to 0:

(Finding $x^\star$) Calculate derivative wrt $x$ and set to 0:

$$\frac{\mathrm{d}}{\mathrm{d}x}\sum_{n=1}^{N}\Big(c_n \log x + (1-c_n)\log(1-x)\Big) = 0$$

$$\sum_{n=1}^{N}\left(\frac{c_n}{x} - \frac{(1-c_n)}{(1-x)}\right) = 0$$

$\frac{\mathrm{d}}{\mathrm{d}x} f(x) = 0$ gives optimum

and

$$\frac{\mathrm{d}}{\mathrm{d}x}\log x = \frac{1}{x}$$

# TOSS A COIN... 🎶

(Finding $x^\star$) Calculate derivative wrt $x$ and set to 0:

$$\frac{\mathrm{d}}{\mathrm{d}x}\sum_{n=1}^{N}\Big(c_n \log x + (1-c_n)\log(1-x)\Big) = 0$$

$$\sum_{n=1}^{N}\left(\frac{c_n}{x} - \frac{(1-c_n)}{(1-x)}\right) = 0$$

$$\sum_{n=1}^{N}\Big(c_n(1-x) - (1-c_n)x\Big) = 0$$

$$\sum_{n=1}^{N} c_n - x\sum_{n=1}^{N} c_n - Nx + x\sum_{n=1}^{N} c_n = 0 \quad \Rightarrow \quad \boxed{x = \frac{1}{N}\sum_{n=1}^{N} c_n}$$

$\frac{\mathrm{d}}{\mathrm{d}x} f(x) = 0$ gives optimum

and

$$\frac{\mathrm{d}}{\mathrm{d}x}\log x = \frac{1}{x}$$

(Finding $x^\star$) Calculate derivative wrt $x$ and set to 0:

$$\frac{\mathrm{d}}{\mathrm{d}x}\sum_{n=1}^{N}\Big(c_n \log x + (1-c_n)\log(1-x)\Big) = 0$$

$$\sum_{n=1}^{N}\left(\frac{c_n}{x} - \frac{(1-c_n)}{(1-x)}\right) = 0$$

$$\sum_{n=1}^{N}\Big(c_n(1-x) - (1-c_n)x\Big) = 0$$

$$\sum_{n=1}^{N} c_n - x\sum_{n=1}^{N} c_n - Nx + x\sum_{n=1}^{N} c_n = 0 \quad \Rightarrow \quad \boxed{x = \frac{1}{N}\sum_{n=1}^{N} c_n}$$

$\frac{\mathrm{d}}{\mathrm{d}x}f(x) = 0$ gives optimum

and

$\frac{\mathrm{d}}{\mathrm{d}x}\log x = \frac{1}{x}$

**EXAMPLE:**

$\mathscr{D} = \{0,0,1,1,0,1,1\}$

$x^\star = 4/7$

# PROBABILISTIC LEARNING (LIKELIHOOD-BASED)

1) Determine $p(y|x)$.

# PROBABILISTIC LEARNING (LIKELIHOOD-BASED)

1) Determine $p(y \mid x)$.

2) Determine $p(\mathscr{D} \mid x)$.

# PROBABILISTIC LEARNING (LIKELIHOOD-BASED)

1) Determine $p(y \mid x)$.

2) Determine $p(\mathscr{D} \mid x)$.

3) Check constraints.

1) Determine $p(y\,|\,x)$.

2) Determine $p(\mathscr{D}\,|\,x)$.

3) Check constraints.

4) Find the best solution by minimizing $-\log p(\mathscr{D}\,|\,x)$.

# PROBABILISTIC LEARNING (LIKELIHOOD-BASED)

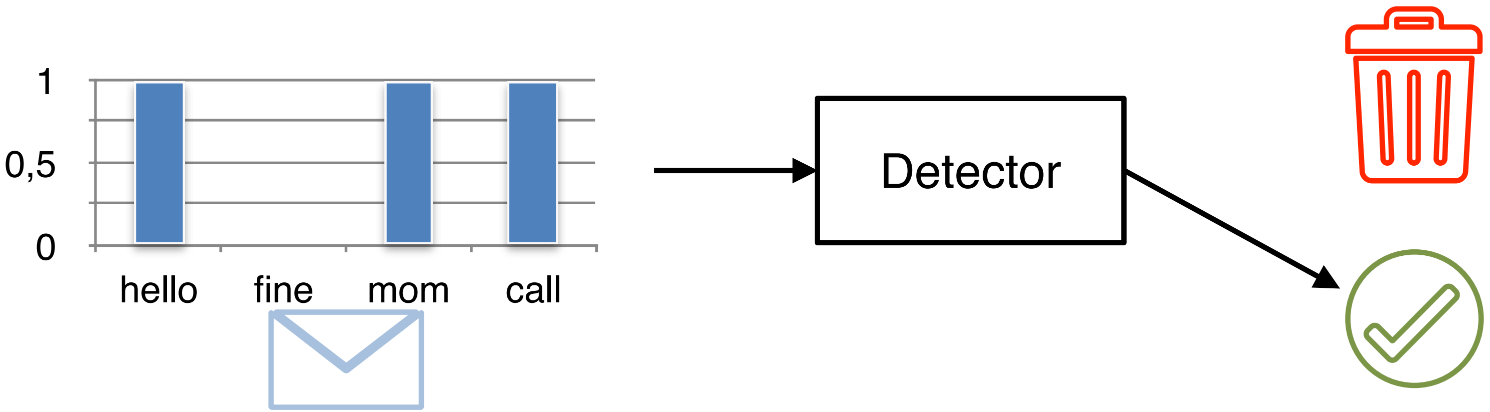
1) Determine $p(y \mid x)$. e.g., Bernoulli, Gaussian…

2) Determine $p(\mathscr{D} \mid x)$. e.g., *iid* or sequential

3) Check constraints. e.g., only values between [0, 1]

4) Find the best solution by minimizing $-\log p(\mathscr{D} \mid x)$. e.g., using gradient-descent

- Example: Spam detection
  - $x \in \{0,1\}^D$ - whether a *d*th word occurs in an e-mail ($x = 1$) or not
  - $y \in \{0,1\}$ - whether an e-mail is a spam ($y = 1$) or not
  - Goal: provide probability of a spam OR classify messages.



- $x \in \mathbb{R}^D$, $y \in \{0,1\}$, $\theta \in \mathbb{R}^D$
- We model $y$ by using the Bernoulli distribution:

$$p(y \mid x, \theta) = \mathrm{Bern}(y \mid \mathrm{sigm}(\theta^\top x))$$

# LOGISTIC REGRESSION

- $x \in \mathbb{R}^D$, $y \in \{0,1\}$, $\theta \in \mathbb{R}^D$
- We model $y$ by using the Bernoulli distribution:

$$p(y \mid x, \theta) = \mathrm{Bern}(y \mid \mathrm{sigm}(\theta^\top x))$$

where:

linear dependency: $\theta^\top x = \sum_{d=1}^{D} \theta_d x_d$

sigmoid function: $\mathrm{sigm}(s) = \dfrac{1}{1 + \exp(-s)}$

- $x \in \mathbb{R}^D$, $y \in \{0,1\}$, $\theta \in \mathbb{R}^D$
- We model $y$ by using the Bernoulli distribution:

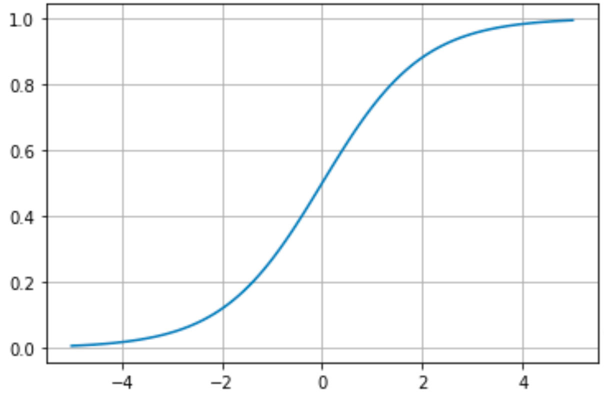
$$p(y\,|\,x, \theta) = \mathrm{Bern}(y\,|\,\mathrm{sigm}(\theta^\top x))$$

where:

linear dependency: $\theta^\top x = \sum_{d=1}^{D} \theta_d x_d$

sigmoid function: $\mathrm{sigm}(s) = \dfrac{1}{1 + \exp(-s)}$

**Sigmoid can model probabilities!**



# LOGISTIC REGRESSION

- Properties of the sigmoid function:
  - $\mathrm{sigm}(s) \in [0,1]$
  - $\dfrac{\mathrm{d}}{\mathrm{d}s}\mathrm{sigm}(s) = \mathrm{sigm}(s)\ (1 - \mathrm{sigm}(s))$
  - $\mathrm{sigm}(-s) = 1 - \mathrm{sigm}(s)$
- In our model we have:

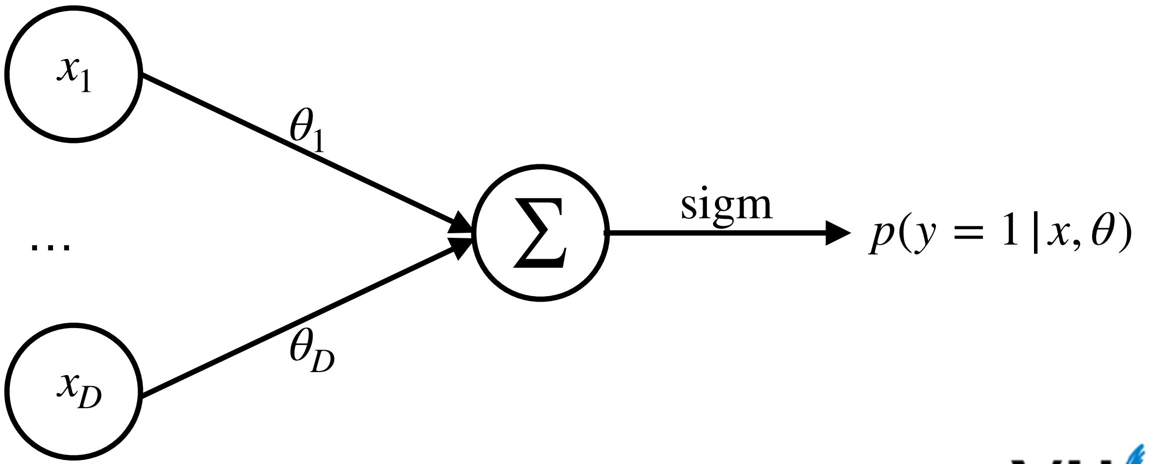
$$p(y = 1 \,|\, x, \theta) = \mathrm{sigm}(\theta^\top x)$$

$$p(y = 0 \,|\, x, \theta) = 1 - \mathrm{sigm}(\theta^\top x)$$

# LOGISTIC REGRESSION

- $p(y|x,\theta) = \mathrm{Bern}(y|\mathrm{sigm}(\theta^\top x))$

where: $\theta^\top x = \sum_{d=1}^{D} \theta_d x_d$



# LOGISTIC REGRESSION: GRADIENT-DESCENT

$$\nabla_\theta - \log p(\mathscr{D}_y | \mathscr{D}_x, \theta) = \nabla_\theta - \sum_i \log p(y_i | x_i, \theta)$$

# LOGISTIC REGRESSION: GRADIENT-DESCENT

$$\nabla_\theta - \log p(\mathscr{D}_y \,|\, \mathscr{D}_x, \theta) = \nabla_\theta - \sum_i \log p(y_i \,|\, x_i, \theta)$$

$$= \nabla_\theta - \sum_i \left( y_i \log \mathrm{sigm}(\theta^\top x_i) + (1 - y_i) \log \mathrm{sigm}(-\theta^\top x_i) \right)$$

# LOGISTIC REGRESSION: GRADIENT-DESCENT

$$\nabla_\theta - \log p(\mathcal{D}_y | \mathcal{D}_x, \theta) = \nabla_\theta - \sum_i \log p(y_i | x_i, \theta)$$

$$= \nabla_\theta - \sum_i \left( y_i \log \mathrm{sigm}(\theta^\top x_i) + (1 - y_i) \log \mathrm{sigm}(-\theta^\top x_i) \right)$$

$$= -\sum_i \left( y_i \frac{1}{\mathrm{sigm}(\theta^\top x_i)} \mathrm{sigm}(\theta^\top x_i)\mathrm{sigm}(-\theta^\top x_i) x_i - (1 - y_i) \frac{1}{\mathrm{sigm}(-\theta^\top x_i)} \mathrm{sigm}(\theta^\top x_i)\mathrm{sigm}(-\theta^\top x_i) x_i \right)$$

# LOGISTIC REGRESSION: GRADIENT-DESCENT

$$\nabla_\theta - \log p(\mathscr{D}_y \mid \mathscr{D}_x, \theta) = \nabla_\theta - \sum_i \log p(y_i \mid x_i, \theta)$$

$$= \nabla_\theta - \sum_i \left( y_i \log \operatorname{sigm}(\theta^\top x_i) + (1 - y_i) \log \operatorname{sigm}(-\theta^\top x_i) \right)$$

$$= - \sum_i \left( y_i \frac{1}{\operatorname{sigm}(\theta^\top x_i)} \operatorname{sigm}(\theta^\top x_i) \operatorname{sigm}(-\theta^\top x_i) x_i - (1 - y_i) \frac{1}{\operatorname{sigm}(-\theta^\top x_i)} \operatorname{sigm}(\theta^\top x_i) \operatorname{sigm}(-\theta^\top x_i) x_i \right)$$

$$= - \sum_i \left( y_i \ \operatorname{sigm}(-\theta^\top x_i) \ x_i - (1 - y_i) \ \operatorname{sigm}(\theta^\top x_i) \ x_i \right)$$

$$= - \sum_i \left( y_i \ \left( \operatorname{sigm}(-\theta^\top x_i) + \operatorname{sigm}(\theta^\top x_i) \right) \ x_i - \operatorname{sigm}(\theta^\top x_i) \ x_i \right)$$

# LOGISTIC REGRESSION: GRADIENT-DESCENT

$$
\begin{aligned}
\nabla_\theta - \log p(\mathcal{D}_y | \mathcal{D}_x, \theta) &= \nabla_\theta - \sum_i \log p(y_i | x_i, \theta) \\
&= \nabla_\theta - \sum_i \left( y_i \log \mathrm{sigm}(\theta^\top x_i) + (1 - y_i) \log \mathrm{sigm}(-\theta^\top x_i) \right) \\
&= - \sum_i \left( y_i \frac{1}{\mathrm{sigm}(\theta^\top x_i)} \mathrm{sigm}(\theta^\top x_i) \mathrm{sigm}(-\theta^\top x_i) x_i - (1 - y_i) \frac{1}{\mathrm{sigm}(-\theta^\top x_i)} \mathrm{sigm}(\theta^\top x_i) \mathrm{sigm}(-\theta^\top x_i) x_i \right) \\
&= - \sum_i \left( y_i \ \mathrm{sigm}(-\theta^\top x_i) \ x_i - (1 - y_i) \ \mathrm{sigm}(\theta^\top x_i) \ x_i \right) \\
&= - \sum_i \left( y_i \ \left( \mathrm{sigm}(-\theta^\top x_i) + \mathrm{sigm}(\theta^\top x_i) \right) \ x_i - \mathrm{sigm}(\theta^\top x_i) \ x_i \right) \\
&= - \sum_i \left( y_i \ x_i - \mathrm{sigm}(\theta^\top x_i) \ x_i \right) \\
&= \sum_i \left( \mathrm{sigm}(\theta^\top x_i) - y_i \right) \ x_i
\end{aligned}
$$

# LOGISTIC REGRESSION: GRADIENT-DESCENT

- The update rule:

$$\theta := \theta - \alpha \sum_{i=1}^{N} \left( \mathrm{sigm}(\theta^\top x_i) - y_i \right) x_i$$

- The update rule:

$$\theta := \theta - \alpha \sum_{i=1}^{N} \left( \mathrm{sigm}(\theta^\top x_i) - y_i \right) x_i$$

- What if $N$ is large?
  - Use mini-batches ($M \ll N$)! (**Stochastic** Gradient descent)

$$\theta := \theta - \alpha \sum_{j=1}^{M} \left( \mathrm{sigm}(\theta^\top x_j) - y_j \right) x_j$$

- The update rule:

$$\theta := \theta - \alpha \sum_{i=1}^{N} \left( \mathrm{sigm}(\theta^\top x_i) - y_i \right) x_i$$

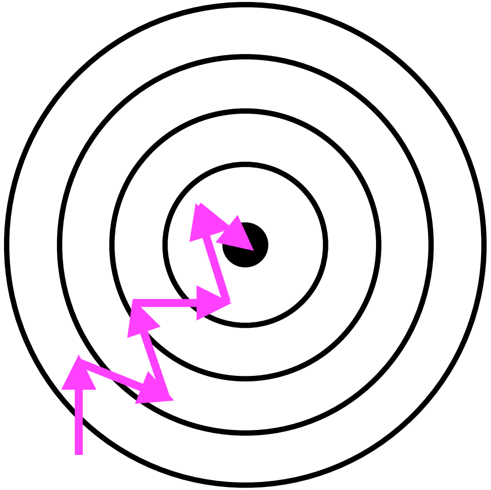
- What if $N$ is large?
  - Use mini-batches ($M \ll N$)! (**Stochastic** Gradient descent)

$$\theta := \theta - \alpha \sum_{j=1}^{M} \left( \mathrm{sigm}(\theta^\top x_j) - y_j \right) x_j$$

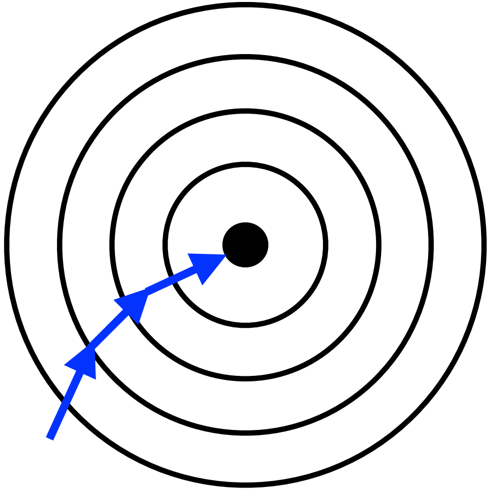
or even:

$$\theta := \theta - \alpha \left( \mathrm{sigm}(\theta^\top x_j) - y_j \right) x_j$$

# LOGISTIC REGRESSION: SGD VS. GD



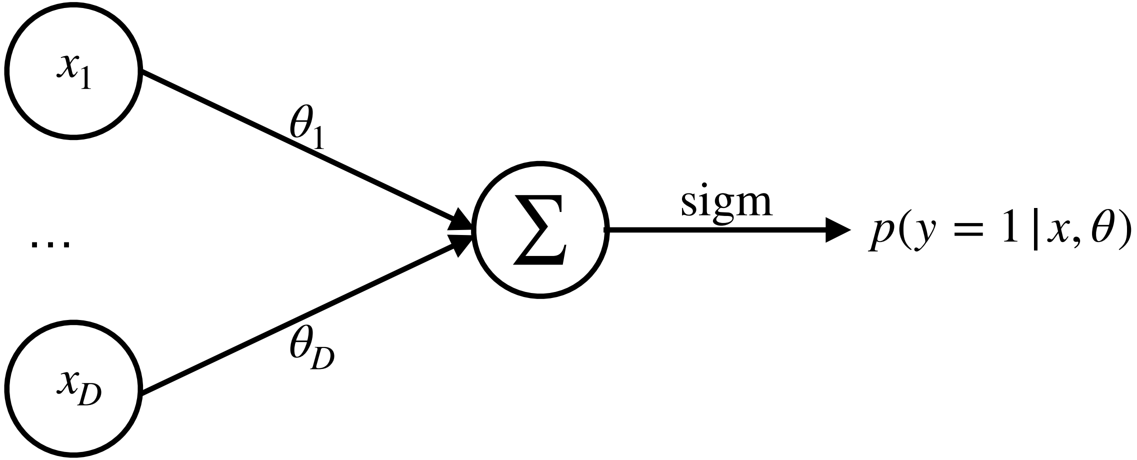
Stochastic Gradient Descent



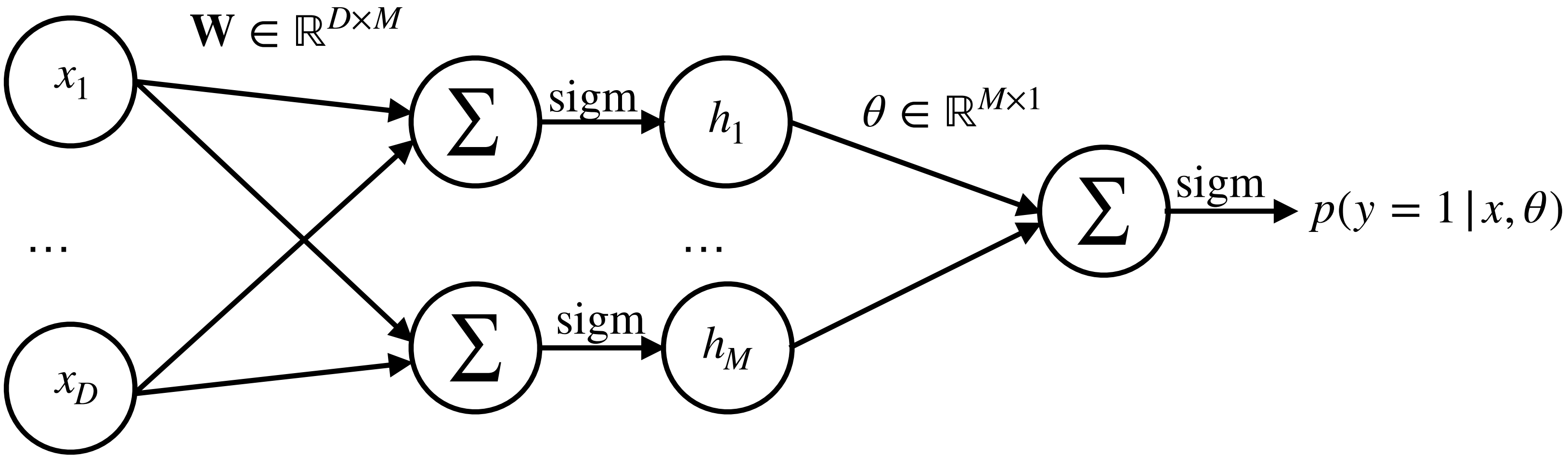
Gradient Descent

# FULLY-CONNECTED NEURAL NETWORKS

# WHAT IF WE STACK MULTIPLE LOGISTIC REGRESSORS



# STACKING LOGISTIC REGRESSORS




**weights**: $\mathbf{W} \in \mathbb{R}^{D \times M}, \ \theta \in \mathbb{R}^{M \times 1}$

# FULLY CONNECTED NEURAL NETWORKS (FCN)

- Stacking logistic regressions models the probability as follows:

$$p(y = 1 \,|\, x, \mathbf{W}, \theta) = \mathrm{sigm}\Big(\theta^\top \underbrace{\mathrm{sigm}(\mathbf{W}x)}_{h}\Big)$$

# FULLY CONNECTED NEURAL NETWORKS (FCN)

- Stacking logistic regressions models the probability as follows:

$$p(y = 1 \,|\, x, \mathbf{W}, \theta) = \mathrm{sigm}\Big(\theta^\top \underbrace{\mathrm{sigm}(\mathbf{W}x)}_{h}\Big)$$

- Notice that we still use the **log-likelihood function** as our **objective**!

# FULLY CONNECTED NEURAL NETWORKS (FCN)

- Stacking logistic regressions models the probability as follows:

$$p(y = 1 \,|\, x, \mathbf{W}, \theta) = \mathrm{sigm}\Big(\theta^\top \underbrace{\mathrm{sigm}(\mathbf{W}x)}_{h}\Big)$$

- Notice that we still use the **log-likelihood function** as our **objective**!
- We refer to $h$ as a **hidden layer**, and $h_m$ is called a **neuron**.

# FULLY CONNECTED NEURAL NETWORKS (FCN)

- Stacking logistic regressions models the probability as follows:

$$p(y = 1 \,|\, x, \mathbf{W}, \theta) = \mathrm{sigm}\Big(\theta^\top \underbrace{\mathrm{sigm}(\mathbf{W}x)}_{h}\Big)$$

- Notice that we still use the **log-likelihood function** as our **objective**!
- We refer to $h$ as a **hidden layer**, and $h_m$ is called a **neuron**.
- We can stack even more:

$$p(y = 1 \,|\, x, \{\mathbf{W}_i\}, \theta) = \mathrm{sigm}\Big(\theta^\top \mathrm{sigm}(\cdots \mathbf{W}_2 \mathrm{sigm}(\mathbf{W}_1 x))\Big)$$

# FCN: COMPONENTS

- A **linear layer**: $a = \mathbf{W}x$ (or with explicit bias: $a = \mathbf{W}x + b$)
- with a **non-linearity**: $h = f(\mathbf{W}x)$

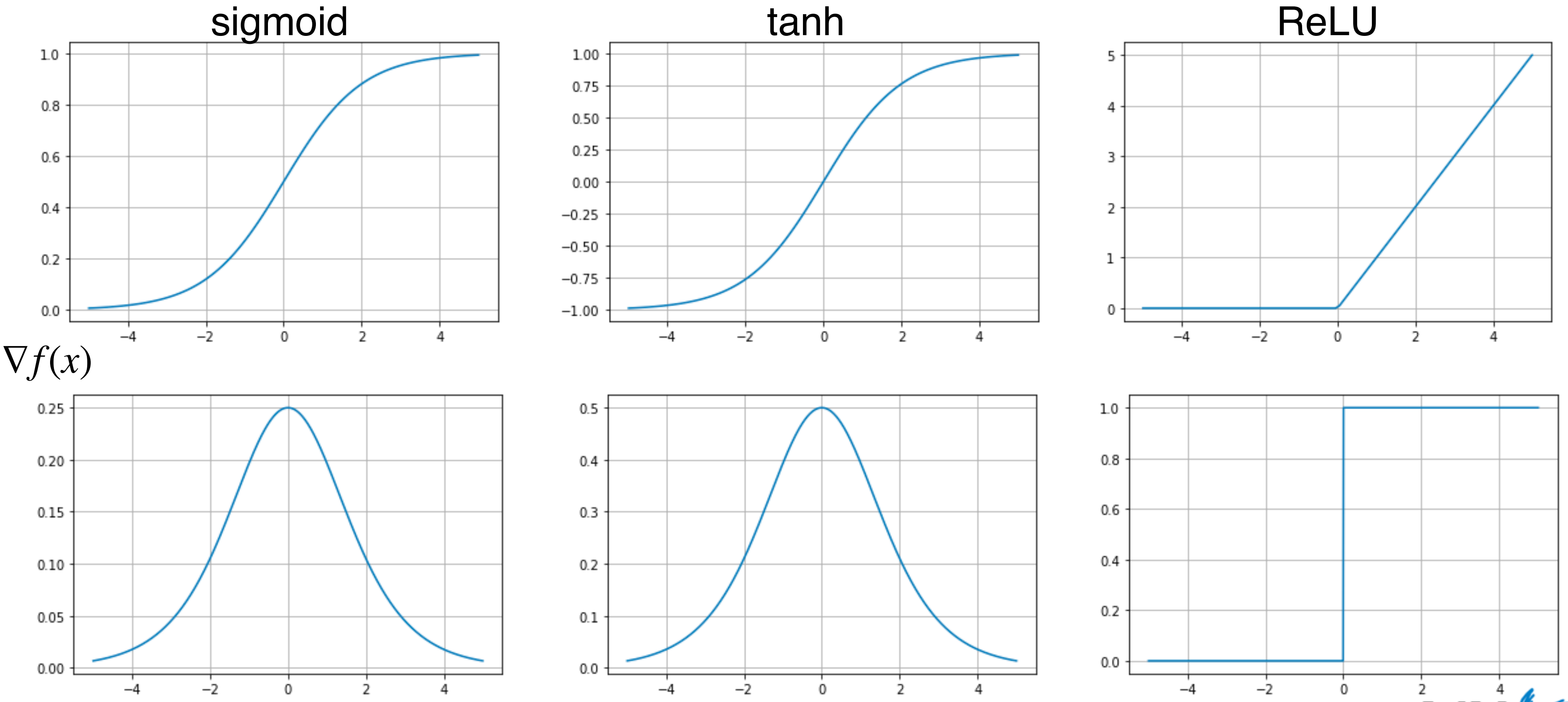
# FCN: COMPONENTS

- A **linear layer**: $a = \mathbf{W}x$ (or with explicit bias: $a = \mathbf{W}x + b$)
- with a **non-linearity**: $h = f(\mathbf{W}x)$
- Typical non-linearities:
  - **sigmoid**: $\mathrm{sigm}(x) = \left(1 + \exp(-x)\right)^{-1}$
  - **tanh**: $\tanh(x) = 2\mathrm{sigm}(2x) - 1$
  - **ReLU**: $\mathrm{relu}(x) = \max\{0, x\}$
  - **softmax**: $\mathrm{softmax}(x) = \exp(x_i) \Big/ \sum \exp(x_j)$

# LINEAR LAYERS

- A **linear layer**: $a = \mathbf{W}x$
- Initialization of $\mathbf{W} \in \mathbb{R}^{D \times M}$:
  - Gaussian: $\mathbf{W} \sim \mathcal{N}(0, \sigma^2)$
  - Xavier (for tanh): $\mathbf{W} \sim \mathcal{N}(0, \sqrt{1/D})$
  - He (for ReLU): $\mathbf{W} \sim \mathcal{N}(0, \sqrt{2/D})$

# ACTIVATION FUNCTIONS



# HOW TO DEAL WITH MULTIPLE CLASSES?

- So far, we talked about 2 classes (sigmoid for modeling probabilities).
- How to deal with $K$ classes?

# HOW TO DEAL WITH MULTIPLE CLASSES?

- So far, we talked about 2 classes (sigmoid for modeling probabilities).
- How to deal with *K* classes?
- **Softmax** function:

$$p(y = i \mid x, \theta) = \frac{\exp(\theta_i^\top x)}{\sum_{k=1}^{K} \exp(\theta_k^\top x)}$$

# LEARNING: BACKPROPAGATION

- We introduced neural networks as stacked logistic regressors.
- How to learn FCN? Can we use SGD?

# LEARNING: BACKPROPAGATION

- We introduced neural networks as stacked logistic regressors.
- How to learn FCN? Can we use SGD? **YES**!

- We introduced neural networks as stacked logistic regressors.
- How to learn FCN? Can we use SGD? **YES**!
- Let us consider one hidden layer:

$$p(y = 1 \,|\, x, \mathbf{W}, \theta) = \mathrm{sigm}\Big(\theta^\top f(\mathbf{W}x)\Big)$$

# LEARNING: BACKPROPAGATION

- We introduced neural networks as stacked logistic regressors.
- How to learn FCN? Can we use SGD? **YES**!
- Let us consider one hidden layer:

$$p(y = 1 \,|\, x, \mathbf{W}, \theta) = \mathrm{sigm}\Big(\theta^\top f(\mathbf{W}x)\Big)$$

- We can use the chain rule to calculate gradients wrt all weights:

$$\frac{\mathrm{d}\ell}{\mathrm{d}u} = \frac{\mathrm{d}\ell}{\mathrm{d}f_1}\frac{\mathrm{d}f_1}{\mathrm{d}f_2}\frac{\mathrm{d}f_2}{\mathrm{d}u}$$

- We introduced neural networks as stacked logistic regressors.
- How to learn FCN? Can we use SGD? **YES**!
- Let us consider one hidden layer:

$$p(y = 1 \,|\, x, \mathbf{W}, \theta) = \mathrm{sigm}\Big(\theta^\top f(\mathbf{W}x)\Big)$$

- We can use the chain rule to calculate gradients wrt all weights:

Negative log-likelihood

$$\frac{\mathrm{d}\ell}{\mathrm{d}u} = \frac{\mathrm{d}\ell}{\mathrm{d}f_1}\frac{\mathrm{d}f_1}{\mathrm{d}f_2}\frac{\mathrm{d}f_2}{\mathrm{d}u}$$

- We introduced neural networks as stacked logistic regressors.
- How to learn FCN? Can we use SGD? **YES**!
- Let us consider one hidden layer:

$$p(y = 1 \,|\, x, \mathbf{W}, \theta) = \mathrm{sigm}\Big(\theta^\top f(\mathbf{W}x)\Big)$$

- We can use the chain rule to calculate gradients wrt all weights:

$$\frac{\mathrm{d}\ell}{\mathrm{d}u} = \frac{\mathrm{d}\ell}{\mathrm{d}f_1}\frac{\mathrm{d}f_1}{\mathrm{d}f_2}\frac{\mathrm{d}f_2}{\mathrm{d}u}$$

**Full derivation:**
**Homework 🤓**
**Or wait till the next lecture!**

# VANISHING GRADIENT PROBLEM

- The sigmoid function is a good choice to model probabilities, however, it is bad as a non-linearity for hidden layers.
- The problem arises while calculating gradients:

$$\frac{\mathrm{d}}{\mathrm{d}s}\mathrm{sigm}(s) = \mathrm{sigm}(s)\ (1 - \mathrm{sigm}(s))$$

If $\mathrm{sigm}(x) \approx 1$, then:

$$\frac{\mathrm{d}}{\mathrm{d}s}\mathrm{sigm}(s) \approx 1\ (1 - 1) = 0$$

# VANISHING GRADIENT PROBLEM

- The sigmoid function is a good choice to model probabilities, however, it is bad as a non-linearity for hidden layers.
- The problem arises while calculating gradients:

$$\frac{\mathrm{d}}{\mathrm{d}s}\mathrm{sigm}(s) = \mathrm{sigm}(s)\ (1 - \mathrm{sigm}(s))$$

If $\mathrm{sigm}(x) \approx 1$, then:

$$\frac{\mathrm{d}}{\mathrm{d}s}\mathrm{sigm}(s) \approx 1\ (1 - 1) = 0$$

**For instance:**

$$\frac{\mathrm{d}\ell}{\mathrm{d}u} = \frac{\mathrm{d}\ell}{\mathrm{d}f_1} \cdot 0 \cdot \frac{\mathrm{d}f_2}{\mathrm{d}u} = 0$$

# Thank you!

# EXTRA READING

Bishop, “Pattern Recognition and Machine Learning”

Murphy, “Machine Learning: A Probabilistic Perspective”

Courville, Goodfellow, Bengio,“Deep Learning”

Kruse et al., “Computational Intelligence: A Methodological Introduction”, Springer