Statistical Machine Learning

Lecture 9 – Deep learning and neural networks

Niklas Wahlström
Division of Systems and Control
Department of Information Technology
Uppsala University

Email: niklas.wahlstrom@it.uu.se

UPPSALA UNIVERSITET

Practical info

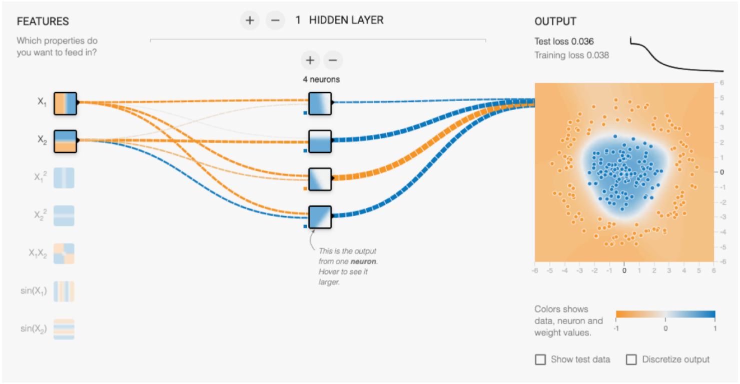
1. **The laboratory work**
    - Format: 4h mandatory computer lab about deep learning. Approved on spot, no report.
    - Three available slots: Monday 8/3 8:15-12:00, Wednesday 10/3 8:15-12:00 and Wednesday 10/3 13:15 - 17:00
    - You do the lab in your mini-project groups
    - **Sign up in Studium**
    - Lab conducted in zoom, each group will be assigned a breakout room
    - Lab-pm and code available in Studium.
    - Do the preparatory exercises **before** the lab session and **submit** your answers in Studium no later than March 7.
    - Install required packackes before lab (PyTorch) or run the lab using Google Colab.
2. **Peer-review of mini-project reports**
    - Deadline Monday 1/3, i.e., next Monday.

UPPSALA UNIVERSITET

Contents – Lecture 9

1. **This lecture:** The neural network model
    - Neural network for regression
    - Neural network for classification
2. **Next lecture:**
    - Convolutional neural network
    - How to train a neural network

Some examples of Neural networks from Lecture 7



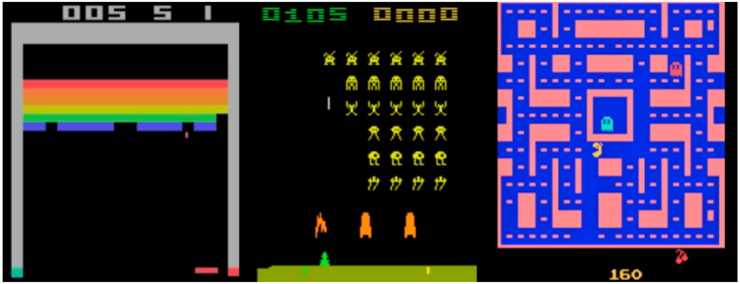
N. Wahlström, 2020

UPPSALA UNIVERSITET

Some examples of Neural networks from Lecture 7

UPPSALA UNIVERSITET

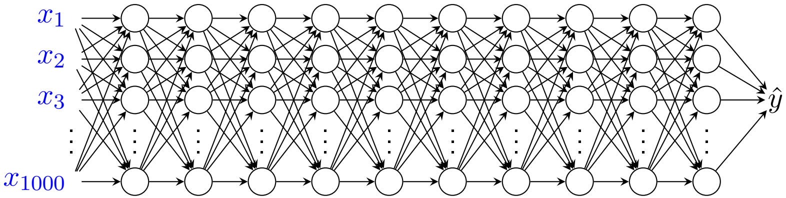
Some examples of Neural networks from Lecture 7



N. Wahlström, 2020

Where are we heading?

We will spend the next 30 minutes deriving the neural network, which graphically can be depicted as the one below



Constructing neural networks for regression

A **neural network (NN)** is a nonlinear function $\hat{y} = f(\mathbf{x}; \boldsymbol{\theta})$ from an input $\mathbf{x}$ to an output $\hat{y}$ parameterized by parameters $\boldsymbol{\theta}$.

Linear regression models the relationship between a continuous output y and a continuous input $\mathbf{x}$,

$$\hat{y} = \sum_{j=1}^{p} W_j x_j + b = \mathbf{W}\mathbf{x} + b,$$

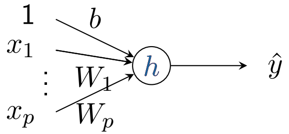
where the parameters $\boldsymbol{\theta}$ are the **weights** W_j, and the **offset** (aka bias/intercept) term b,

$$\boldsymbol{\theta} = \begin{bmatrix} b & \mathbf{W} \end{bmatrix}^{\mathsf{T}}, \qquad \mathbf{W} = \begin{bmatrix} W_1 & W_2 & \cdots & W_p \end{bmatrix}$$

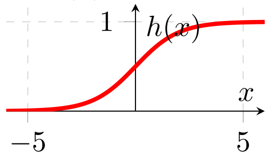
Generalized linear regression

We can generalize this by introducing nonlinear transformations of the predictor $\mathbf{W}\mathbf{x} + b$,

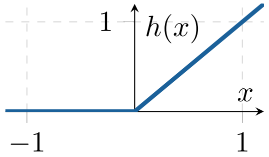
$$\hat{y} = h(\mathbf{W}\mathbf{x} + b),$$



We call $h(x)$ the **activation function**. Two common choices are:



Sigmoid: $h(x) = \frac{e^x}{1+e^x}$

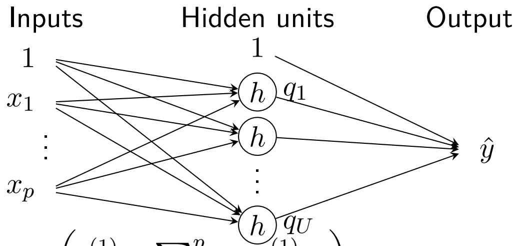


ReLU: $h(x) = \max(0, x)$

Let us consider an example of a **neural network**.

Neural network - construction

A neural network is a sequential construction of **several** generalized linear regression models.



$$q_1 = h\left(b_1^{(1)} + \sum_{j=1}^{p} W_{1j}^{(1)} x_j\right)$$

$$q_2 = h\left(b_2^{(1)} + \sum_{j=1}^{p} W_{2j}^{(1)} x_j\right)$$

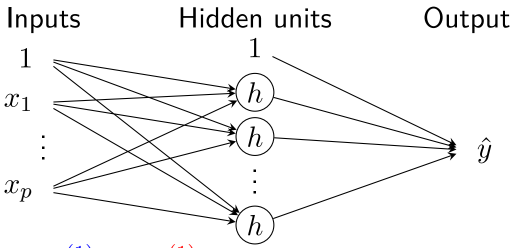
$$\vdots$$

$$q_U = h\left(b_U^{(1)} + \sum_{j=1}^{p} W_{Uj}^{(1)} x_j\right)$$

$$\hat{y} = b^{(2)} + \sum_{i=1}^{U} W_i^{(2)} q_i$$

Neural network - construction

A neural network is a sequential construction of **several** generalized linear regression models.



$$\mathbf{q} = h(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) \qquad \hat{y} = \mathbf{W}^{(2)}\mathbf{q} + \mathbf{b}^{(2)}$$

$$\mathbf{W}^{(1)} = \begin{bmatrix} W_{11}^{(1)} & \dots & W_{1p}^{(1)} \\ \vdots & & \vdots \\ W_{U1}^{(1)} & \dots & W_{Up}^{(1)} \end{bmatrix}, \quad \mathbf{b}^{(1)} = \begin{bmatrix} b_1^{(1)} \\ \vdots \\ b_U^{(1)} \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} q_1 \\ \vdots \\ q_U \end{bmatrix}$$

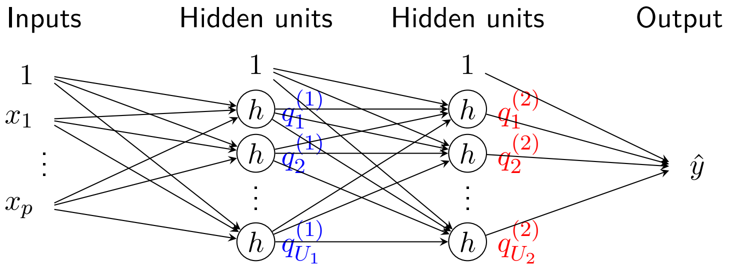
Weight matrix — Offset vector — Hidden units

$$\mathbf{b}^{(2)} = \begin{bmatrix} b^{(2)} \end{bmatrix}$$

$$\mathbf{W}^{(2)} = \begin{bmatrix} W_1^{(2)} & \dots & W_U^{(2)} \end{bmatrix}$$

Neural network - construction

A neural network is a **sequential** construction of several generalized linear regression models.



$$\mathbf{q}^{(1)} = h(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$
$$\mathbf{q}^{(2)} = h(\mathbf{W}^{(2)}\mathbf{q}^{(1)} + \mathbf{b}^{(2)})$$
$$\hat{y} = \mathbf{W}^{(3)}\mathbf{q}^{(2)} + \mathbf{b}^{(3)}$$

The model learns better using a deep network (several layers) instead of a wide and shallow network.

UPPSALA UNIVERSITET

Deep learning

A neural network with L layers can be written as

$$\begin{aligned} \mathbf{q}^{(0)} &= \mathbf{x} \\ \mathbf{q}^{(l)} &= h\left(\mathbf{W}^{(l)}\mathbf{q}^{(l-1)} + \mathbf{b}^{(l)}\right), \quad l = 1, \dots, L-1 \\ \hat{y} &= \mathbf{W}^{(L)}\mathbf{q}^{(L-1)} + \mathbf{b}^{(L)} \end{aligned}$$

All weight matrices and offset vectors in all layers combined are the parameters of the network

$$\boldsymbol{\theta} = \left[\mathrm{vec}(\mathbf{W}^{(1)})^{\mathsf{T}}, \;\; \mathbf{b}^{(1)\mathsf{T}}, \;\; \dots, \;\; \mathrm{vec}(\mathbf{W}^{(L)})^{\mathsf{T}}, \;\; \mathbf{b}^{(L)\mathsf{T}}\right]^{\mathsf{T}},$$

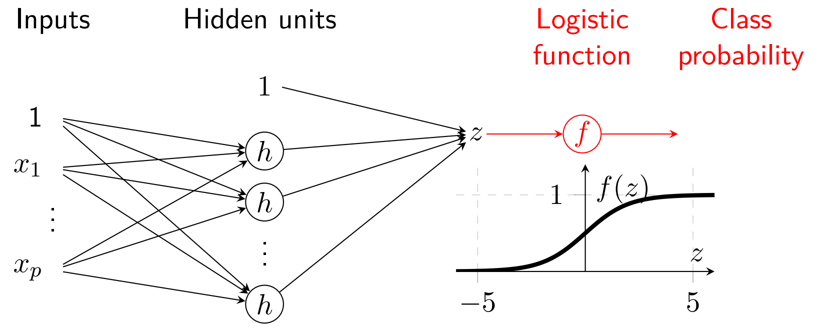
which constitutes the parametric model $\hat{y} = f(\mathbf{x}; \boldsymbol{\theta})$. If L is large we call it a deep neural network.

Deep learning is a class of machine learning models and algorithms that use a cascade of multiple layers, each of which is a nonlinear transformation.

NN for classification ($M = 2$ classes)

We can also use neural networks for classification. With $M = 2$ classes we squash the output through the logistic function to get a **class probability** $f(z) \in [0, 1]$

$$f(z) = p(y = 1|\mathbf{x}, \boldsymbol{\theta}) \quad \text{where} \quad f(z) = \frac{e^z}{1 + e^z}$$

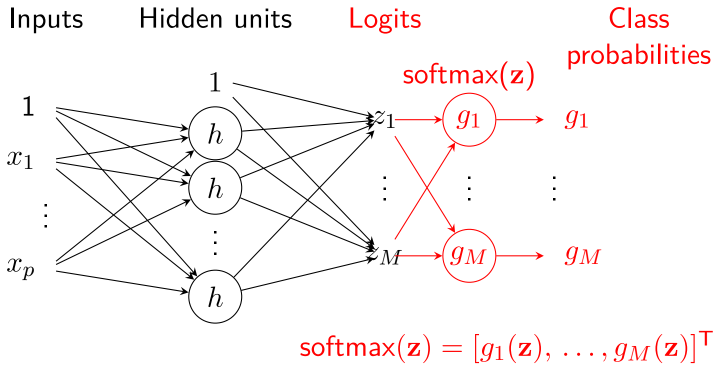


What if $M > 2$?

NN for classification ($M > 2$ classes)

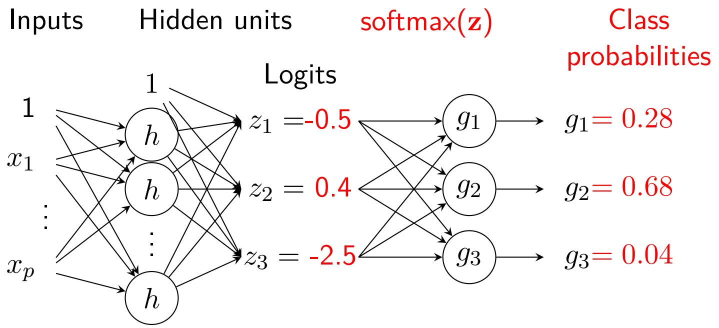
For $M > 2$ classes we want to predict the class probability for all M classes $g_m(\mathbf{z}) = p(y = m|\mathbf{x}, \boldsymbol{\theta})$. We extend the logistic function to the **softmax function**

$$g_m(\mathbf{z}) = \frac{e^{z_m}}{\sum_{l=1}^{M} e^{z_l}}, \qquad m = 1, \ldots, M.$$



NN classification with $M = 3$ classes (I/II)

Consider an example with three classes $M = 3$.



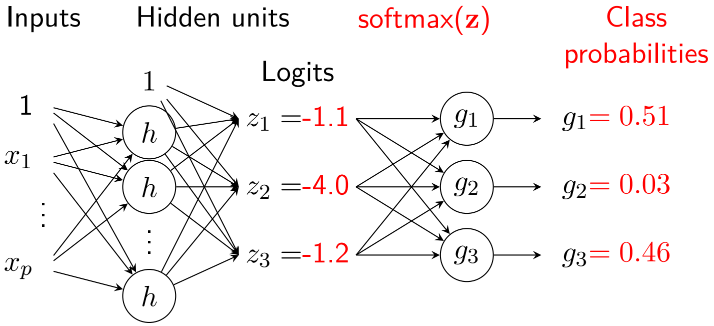
where

$$g_m(\mathbf{z}) = \frac{e^{z_m}}{\sum_{l=1}^{M} e^{z_l}}, \qquad \text{softmax}(\mathbf{z}) = [g_1(\mathbf{z}), \ldots, g_M(\mathbf{z})]^\mathsf{T}.$$

UPPSALA UNIVERSITET

NN classification with $M = 3$ classes (I/II)

Consider an example with three classes $M = 3$.



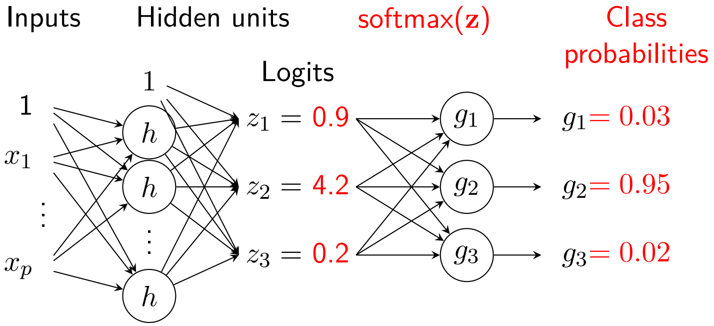
where

$$g_m(\mathbf{z}) = \frac{e^{z_m}}{\sum_{l=1}^{M} e^{z_l}}, \qquad \text{softmax}(\mathbf{z}) = [g_1(\mathbf{z}), \dots, g_M(\mathbf{z})]^{\mathsf{T}}.$$

UPPSALA UNIVERSITET

NN classification with $M = 3$ classes (I/II)

Consider an example with three classes $M = 3$.



where

$$g_m(\mathbf{z}) = \frac{e^{z_m}}{\sum_{l=1}^{M} e^{z_l}}, \qquad \text{softmax}(\mathbf{z}) = [g_1(\mathbf{z}), \dots, g_M(\mathbf{z})]^\mathsf{T}.$$

One-hot encoding (for $M > 2$)

We use **one-hot encoding** for the output $\tilde{\mathbf{y}} = [\tilde{y}_1, \dots, \tilde{y}_M]^\mathsf{T}$, which means

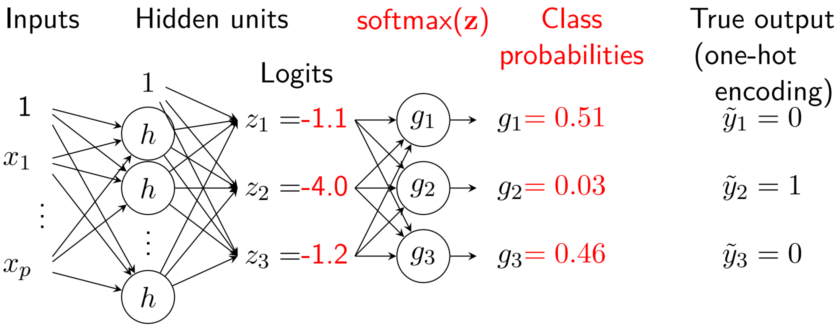
$$\tilde{y}_m = \begin{cases} 1 & \text{if } y = m \\ 0 & \text{if } y \neq m \end{cases}$$

Example: We have $M = 3$ classes and if belongs to ...

- ...$y = 1$, then $\tilde{\mathbf{y}} = [1, \ 0, \ 0]^\mathsf{T}$.
- ...$y = 2$, then $\tilde{\mathbf{y}} = [0, \ 1, \ 0]^\mathsf{T}$.
- ...$y = 3$, then $\tilde{\mathbf{y}} = [0, \ 0, \ 1]^\mathsf{T}$.

NN classification with $M = 3$ classes (II/II)

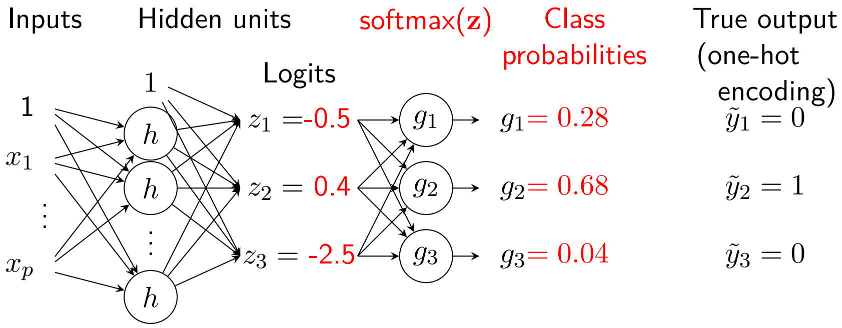
Consider an example with three classes $M = 3$ where $y = 2$.



predicted class probabilities $\leftrightarrow$ true output

NN classification with $M = 3$ classes (II/II)

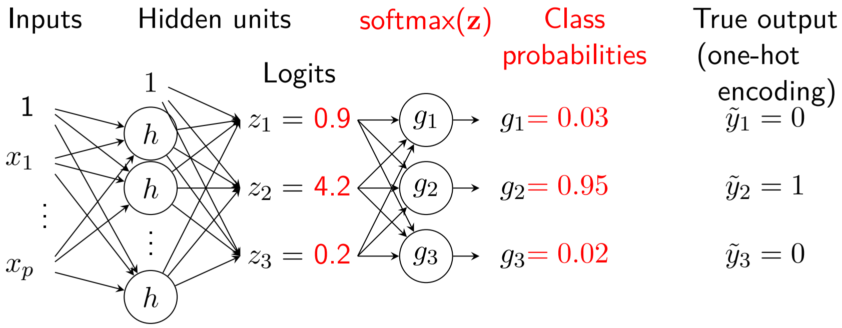
Consider an example with three classes $M = 3$ where $y = 2$.



predicted class probabilities $\leftrightarrow$ true output

NN classification with $M = 3$ classes (II/II)

Consider an example with three classes $M = 3$ where $y = 2$.



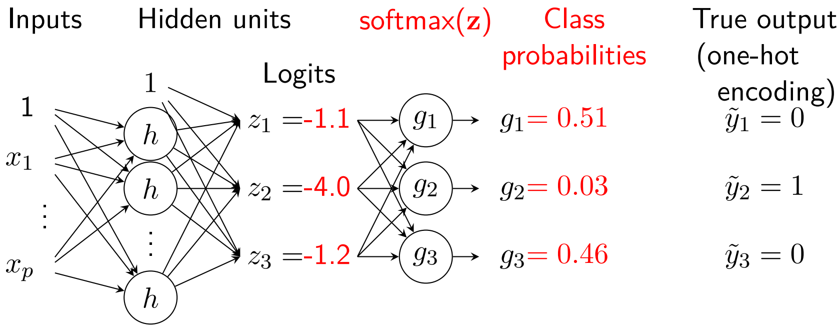
predicted class probabilities $\leftrightarrow$ true output

Q:Which loss do we use between g and $\tilde{y}$?

A: Derive using maximum likelihood! $\Rightarrow$ Cross-entropy loss

NN classification with $M = 3$ classes (II/II)

Consider an example with three classes $M = 3$ where $y = 2$.

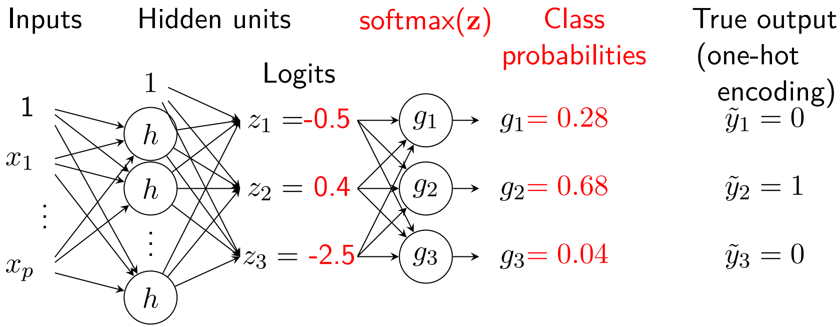


The network is trained by minimizing the **cross-entropy**

$$L(\tilde{\mathbf{y}}, \mathbf{g}) = -\sum_{m=1}^{M} \tilde{y}_m \ln(g_m) = -\ln 0.03 = 3.51$$

NN classification with $M = 3$ classes (II/II)

Consider an example with three classes $M = 3$ where $y = 2$.



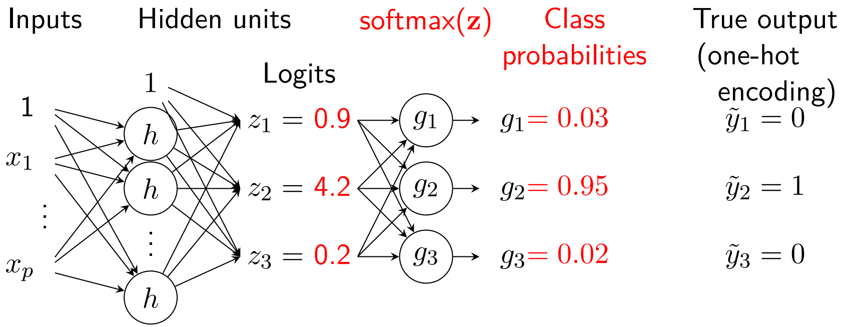
The network is trained by minimizing the **cross-entropy**

$$L(\tilde{\mathbf{y}}, \mathbf{g}) = -\sum_{m=1}^{M} \tilde{y}_m \ln(g_m) = -\ln 0.68 = 0.39$$

UPPSALA UNIVERSITET

NN classification with $M = 3$ classes (II/II)

Consider an example with three classes $M = 3$ where $y = 2$.



The network is trained by minimizing the **cross-entropy**

$$L(\tilde{\mathbf{y}}, \mathbf{g}) = -\sum_{m=1}^{M} \tilde{y}_m \ln(g_m) = -\ln 0.95 = 0.05$$

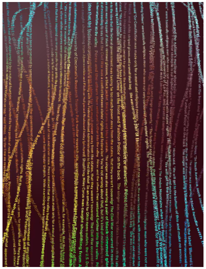
UPPSALA UNIVERSITET

Example: Language models

One result on the use of deep learning for language modeling

Language Models - February 2019
Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.
Language Models are Unsupervised Multitask Learners.,

February, 2019.



See also blog post:
`https://blog.openai.com/better-language-models/`

Language models - Background

A language model is a probabilistic model over sequences of words.

Training

$$\underbrace{\text{This is an introductory course to statistical machine}}_{\text{input}}\ \underbrace{\text{learning}}_{\text{output}}$$

The model can be used to hallucinate new texts.

Prediction

$$\underbrace{\text{Machine learning is the scientific study of algorithms and}}_{\text{input}}\ \underbrace{\text{statistical}}_{\text{prediction}}$$

In the paper they trained a huge language model with 1.5 billion (!) parameters using 40 GB of text data from internet.

Can also be used for Reading Comprehension, Language Translation, Question Answering etc...

Language models - Synthetic text (I/IV)

System Prompt (human-written) *In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.*

Model Completion (machine-written)
The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

...

Language models - Synthetic text (II/IV)

...

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez.

...

Language models - Synthetic text (III/IV)

...

Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them - they were so close they could touch their horns.

While examining these bizarre creatures the scientists discovered that the creatures also spoke some fairly regular English. Pérez stated, "We can see, for example, that they have a common 'language,' something like a dialect or dialectic."

Dr. Pérez believes that the unicorns may have originated in Argentina, where the animals were believed to be descendants of a lost race of people who lived there before the arrival of humans in those parts of South America.

...

UPPSALA UNIVERSITET

Language models - Synthetic text (IV/IV)

...

While their origins are still unclear, some believe that perhaps the creatures were created when a human and a unicorn met each other in a time before human civilization. According to Pérez, ”In South America, such incidents seem to be quite common.”

However, Pérez also pointed out that it is likely that the only way of knowing for sure if unicorns are indeed the descendants of a lost alien race is through DNA. ”But they seem to be able to communicate in English quite well, which I believe is a sign of evolution, or at least a change in social organization,” said the scientist.

Language models - policy implication

Could be used for

- AI writing assistants
- More capable dialogue agents
- Unsupervised translation between languages
- Better speech recognition systems

Potential malicious purposes

- Generate misleading news articles
- Impersonate others online
- Automate the production of faked content on social media
- Automate the production of spam/phishing content

In May 2020 Open AI realised an improved model with 175 bilion (!!!) parameters!

Some comments - Why now?

Neural networks have been around for more than fifty years. Why have they become so popular now (again)?

To solve really interesting problems you need:

1. Efficient learning **algorithms**
2. Efficient computational **hardware**
3. A lot of labeled **data**!

These three factors have not been fulfilled to a satisfactory level until the last 5-10 years.

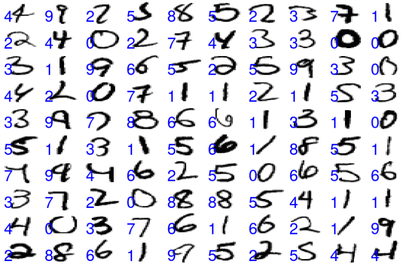
UPPSALA UNIVERSITET

The lab

Topic: Image classification with neural networks

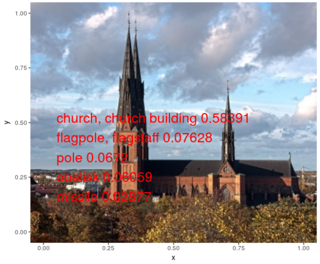
Task 1

Classification of hand-written digits



Task 2

Real world image classification



- Read Section 2 in the lab instruction and do the preparatory exercises in Section 3 **before** the lab

A few concepts to summarize lecture 8

Neural network (NN): A nonlinear parametric model constructed by stacking several linear models with intermediate nonlinear activation functions.

Activation function: (a.k.a squashing function) A nonlinear scalar function applied to each output element of the linear models in a NN.

Hidden units: Intermediate variables in the NN which are not observed, i.e. belongs neither to the input nor output data.

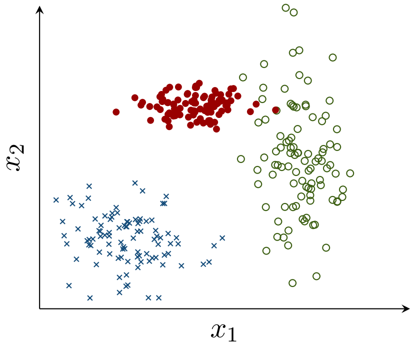
Softmax: A function used to transform the output of a NN to class probabilities.

One-hot encoding: An encoding of the output for training NNs with softmax output.

UPPSALA UNIVERSITET

Softmax regression - A multi-class problem

Consider a classification problem with 3 classes
Input: $\mathbf{x} = [x_1, x_2] \in \mathbb{R}^2$
Output: $y = \{$"red", "blue", "green"$\}$



Can we extend logistic regression to handle more than 2 classes?

Softmax regression

Consider a multi-class classification problem $y \in \{1, \dots, M\}$.

$$g_{im} = p(y_i = m | \mathbf{x}_i)$$

Softmax regression Assume a linear model for each "log odds"

$$\ln \frac{g_{im}}{Z} = \mathbf{w}_m^\mathsf{T} \mathbf{x}_i + b_m, \qquad m = 1, \dots, M$$

where Z is a **normalization factor** such that $\sum_{m=1}^{M} g_{im} = 1$. $\quad \Rightarrow$

$$g_{im} = Z e^{z_{im}}, \quad z_{im} = \mathbf{w}_m^\mathsf{T} \mathbf{x}_i + b_m, \qquad m = 1, \dots, M$$

The normalization factor Z can now be computed as

$$1 = \sum_{m=1}^{M} g_{im} = Z \sum_{m=1}^{M} e^{z_{im}} \quad \Rightarrow \quad Z = \frac{1}{\sum_{m=1}^{M} e^{z_{im}}}$$

Softmax regression model is then

$$g_{im} = \frac{e^{z_{im}}}{\sum_{l=1}^{M} e^{z_{il}}}, \qquad z_{im} = \mathbf{w}_m^\mathsf{T} \mathbf{x}_i + b_m, \qquad m = 1, \dots, M$$

UPPSALA UNIVERSITET

Softmax regression using maximum limelihood

Pick $\mathbf{w}_m$ and b_m which make data as likely as possible

$$\widehat{\mathbf{w}}_{1:M}, \widehat{b}_{1:M} = \underset{\mathbf{w}_{1:M}, b_{1:M}}{\text{argmax}} \; \ln p(y_{1:n} | \mathbf{x}_{1:n}, \mathbf{w}_{1:M}, b_{1:M})$$

Assume all y_i to be independent

$$\begin{aligned}\ln p(y_{1:n} | \mathbf{x}_{1:n}, \mathbf{w}_{1:M}, b_{1:M}) &= \sum_{i=1}^{n} \ln p(y_i | \mathbf{x}_i, \mathbf{w}_{1:M}, b_{1:M}) \\ &= \sum_{m=1}^{M} \sum_{\substack{i=1 \\ \text{where} \\ y_i = m}}^{n} \ln \underbrace{p(y_i = m | \mathbf{x}_i, \mathbf{w}_{1:M}, b_{1:M})}_{= g_{im}}\end{aligned}$$

This leads to the following optimization problem

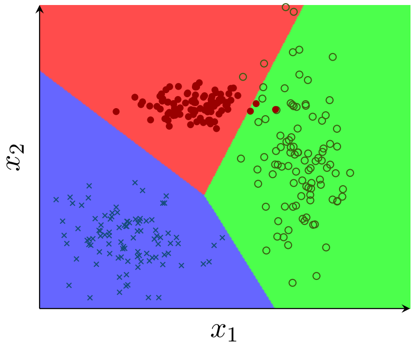
$$\widehat{\mathbf{w}}_{1:M}, \widehat{b}_{1:M} = \underset{\mathbf{w}_{1:M}, b_{1:M}}{\text{argmin}} \; -\sum_{i=1}^{n} \sum_{m=1}^{M} \tilde{y}_{im} \ln(g_{im}),$$

One-hot encoding of output

$$\boxed{\tilde{y}_{im} = \begin{cases} 1, & \text{if} \quad y_i = m \\ 0, & \text{if} \quad y_i \neq m \end{cases}}$$

Softmax regression on multi-class problem

Consider again the classification problem with 3 classes.



Softmax computes the probability for each of the three classes.

Here the result is visualized with **decision boundaries** for the class which has the highest probability.