



Inhaltsverzeichnis

1	Introduction	1
1.1	Overview	1
1.2	Chapter 1.1 - What is machine learning	1
1.3	Chapter 1.2 - What is deep learning	2
1.3.1	What is a neural network (NN)	3
1.4	Chapter 1.3 - Examples for Deep Learning	3
2	Tools for Deep Learning	3
2.1	Datasets	3
3	Machine learning basics	3
3.1	Linear Algebra	4
3.2	Chapter 3.2 Random variable and probability distribution	4
3.2.1	3.2.1 One random vector	4
3.3	Chapter 3.3 - Multiple random vectors	6
3.3.1	Chapter 3.2.3 - Kernel basaed density estimation	6
3.4	Chapter 3.4 Kullback-Leibler divergence and cross entropy	7
3.4.1	E3.5 KLD between normal and Laplace distribution	8
3.5	Chapter 3.5 Probabilistic framework	9
3.5.1	Role of a NN	11
4	Dense Neural Networks	11
4.0.1	4.1 Fully connected neural networks - Neuron	12
4.0.2	Chapter 4.2 Layer of Nurons	12
4.1	4.3 Feedforward neural network	13
5	Dense Neural Networks	15
5.1	Activation function	15
5.1.1	Sigmoid activation function	15
5.1.2	hyperbolic tangent activation function	16
5.1.3	rectifier linear unit(ReLU	17
5.1.4	Softmax activatoin function(classification problem)	17
5.2	Special case c=2, binary classification problem	18
5.3	Chapter 4.5 Universal approximation	18
5.3.1	E4.3 Regression with 1 hidden layer	19
5.4	Chapter 4.4 - 4.6 Loss and cost function	20
5.4.1	4.6.3 Semantic segmentation	21
5.5	Chapter 4.7 Training	22

Abbildungsverzeichnis

1	Introduction	1
1.1	Mathematical notations	1
1.2	Three steps of machine learning	2
1.3	Neural Network	3
2	Tools for Deep Learning	3
3	Machine learning basics	3
3.1	Vector Normes	4
3.2	One random vector	4
3.3	Moments of a vector	5
3.4	Multivariate normal (Gaussian) distribution	5
3.5	Kernel function	6
3.6	Estimated PDF function	7
3.7	Forward vs. Backward KL divergence	8
3.8	Probabilistic framework of supervised learning	9
3.9	The data generating distribution	9
3.10	Bayes Rule in DL	10
3.11	Bayes decision theorem	10
3.12	Supervised learning	10
3.13	Calc part1	11
3.14	Calc part2	11
4	Dense Neural Networks	11
4.1	Neuron	12
4.2	Layer of Neurons	12
4.3	Meanings of phi	13
4.4	Feedforward multilayer neural network	13
4.5	Network Parameters	14
5	Dense Neural Networks	15

List of Equations

1	Introduction	1
2	Tools for Deep Learning	3
3	Machine learning basics	3

4 Dense Neural Networks **11**

5 Dense Neural Networks **15**

Lecture 1: Introduction

Date: 03/05/2020

Lecturer: David Silver

By: Nithish Moudhgalya

1.1. Overview

- p.4 expert talks are not relevant for the exam

- **scalar**: x, A, α

- column **vector**: $\underline{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = [x_i] \in \mathbb{R}^N$ with element $x_i = [\underline{x}]_i$

- **matrix**: $A = [a_{ij}]_{1 \leq i \leq M, 1 \leq j \leq N} = [a_{ij}] \in \mathbb{R}^{M \times N}$ with element $a_{ij} = [A]_{ij}$

- 3D, 4D **tensor**: $A = [a_{ijk}], A = [a_{ijkl}]$

- **transpose** of vector and matrix: $\underline{x}^T = [x_1, \dots, x_N], A^T = [a_{ji}]_{ij}$

- **determinant** of a square matrix A : $|A|$

- **inverse** of a square matrix A : A^{-1}

- **trace** of a square matrix A : $\text{tr}(A) = \sum_i a_{ii}$

Abbildung 1.1: Mathematical notations

- Element notation (element of vector): $[x_i = \underline{a} + \underline{b}]_i$

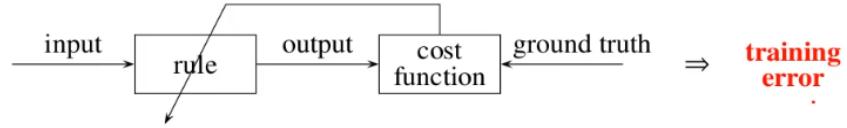
1.2. Chapter 1.1 - What is machine learning

- Signal processing is not a subset of ML or the other way around, they are identical in the task
- Basic difference is in how to design the processing rule
- Regression means to calculate (SP,ML) a continuous-valued output in the real numbers from a signal, can be one-dimensional
- Classification means to calculate (SP,ML) a discrete-valued output from the natural numbers
- p. 1-5 filter in the middle is the view of a pixel and its 8 neighbours

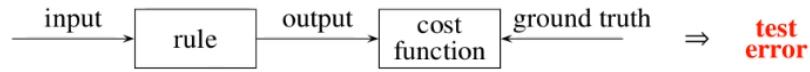
Three steps of machine learning (1)

Dataset $D = D_{\text{train}} \cup D_{\text{test}}$	training set D_{train}	test set D_{test}
---	--	-----------------------------------

a) **Training** using **training data** from the training set D_{train}



b) **Test or evaluation** using **test data** from the test set D_{test}



c) Deployment on new data

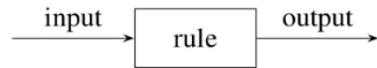


Abbildung 1.2: Three steps of machine learning

- over-fitting just memorizes the training set so test set is needed
- testerrorrate is similar to training error rate → no over-fitting
- Test set and training set need to be disjunct concatenated
- TODO: Summary supervised learning and unsupervised learning

1.3. Chapter 1.2 - What is deep learning

- feature extraction: a feature is a clustered subset of information for recognition
- fish example: fish length, color etc.
- Needs human experience, can't be calculated
- DNN solves the problem without feature extraction

1.3.1. What is a neural network (NN)

- Cascaded pipeline of layers

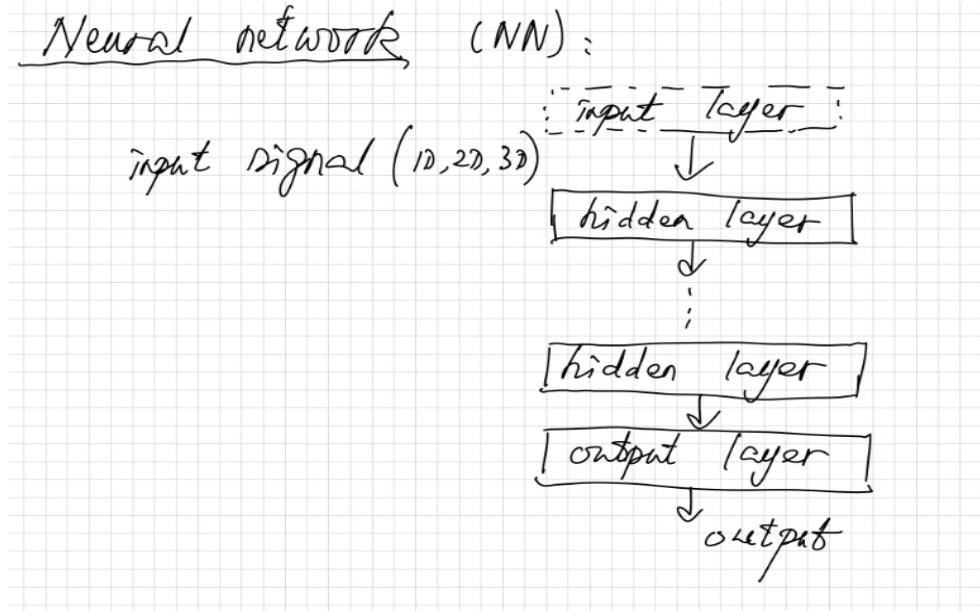


Abbildung 1.3: Neural Network

1.4. Chapter 1.3 - Examples for Deep Learning

- Semantic image segmentation is pixelwise classification

Lecture 2: Tools for Deep Learning

Date: 02/05/2020

Lecturer: David Silver

By: Nithish Moudhgalya

2.1. Datasets

- Overview for Training Datasets up to ImageNet they are for teaching

Lecture 3: Machine learning basics

Date: 02/05/2020

Lecturer: David Silver

By: Nithish Moudhgalya

3.1. Linear Algebra

- TODO get Math nicely into the Context book

3.1 Linear algebra

3-4

Vector norms

Given a vector $\underline{x} = [x_i] \in \mathbb{R}^M$. There are different definitions for the vector norm:

- **2-norm or l_2 -norm or Euclidean norm:** $\|\underline{x}\|_2 = \sqrt{\sum_{i=1}^M x_i^2} = \sqrt{\underline{x}^T \underline{x}}$
- **1-norm or l_1 -norm:** $\|\underline{x}\|_1 = \sum_{i=1}^M |x_i|$
- **0-norm or l_0 -norm:** $\|\underline{x}\|_0 = \text{number of non-zero elements in } \underline{x}$

Comments:

- $\|\underline{x}\|_2^2$ represents the energy of \underline{x} .
- $\|\underline{x}\|_0$ measures the sparsity of \underline{x} .
- Different vector norms have different unit-norm contour lines $\{\underline{x} \mid \|\underline{x}\|_p = 1\}$.

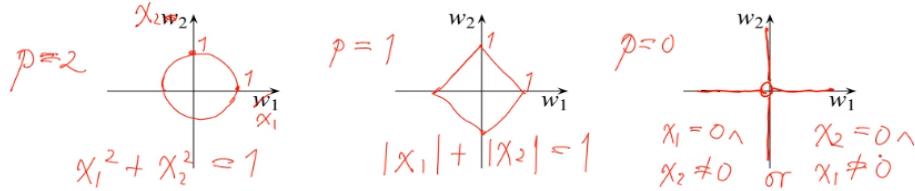


Abbildung 3.1: Vector Normes

3.2. Chapter 3.2 Random variable and probability distribution

3.2.1. 3.2.1 One random vector

- **scalar:** x, A, α

- **column vector:** $\underline{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = [x_i] \in \mathbb{R}^N$ with element $x_i = [\underline{x}]_i$

- **matrix:** $\mathbf{A} = [a_{ij}]_{1 \leq i \leq M, 1 \leq j \leq N} = [a_{ij}] \in \mathbb{R}^{M \times N}$ with element $a_{ij} = [\mathbf{A}]_{ij}$

- **3D, 4D tensor:** $\mathbf{A} = [a_{ijk}], \mathbf{A} = [a_{ijkl}]$

- **transpose** of vector and matrix: $\underline{x}^T = [x_1, \dots, x_N], \mathbf{A}^T = [a_{ji}]_{ij}$

- **determinant** of a square matrix \mathbf{A} : $|\mathbf{A}|$

- **inverse** of a square matrix \mathbf{A} : \mathbf{A}^{-1}

- **trace** of a square matrix \mathbf{A} : $\text{tr}(\mathbf{A}) = \sum_i a_{ii}$

Abbildung 3.2: One random vector

PDF for a discrete-valued RV:

$$p(\underline{x}) = \sum_i p_i \delta(\underline{x} - \underline{x}_i,) \delta(\underline{x} :)$$

cumulative distribution function CCDF

$$F(\underline{x}) = \int_{-\infty}^{\infty} p(\underline{z}) d\underline{z}, \quad p(\underline{x}) = \frac{\partial^d F(\underline{x})}{\partial x_1 \dots \partial x_d}$$

- **scalar**: x, A, α

- column **vector**: $\underline{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = [x_i] \in \mathbb{R}^N$ with element $x_i = [\underline{x}]_i$

- **matrix**: $\mathbf{A} = [a_{ij}]_{1 \leq i \leq M, 1 \leq j \leq N} = [a_{ij}] \in \mathbb{R}^{M \times N}$ with element $a_{ij} = [\mathbf{A}]_{ij}$

- 3D, 4D **tensor**: $\mathbf{A} = [a_{ijk}], \mathbf{A} = [a_{ijkl}]$

- **transpose** of vector and matrix: $\underline{x}^T = [x_1, \dots, x_N], \mathbf{A}^T = [a_{ji}]_{ij}$

- **determinant** of a square matrix \mathbf{A} : $|\mathbf{A}|$

- **inverse** of a square matrix \mathbf{A} : \mathbf{A}^{-1}

- **trace** of a square matrix \mathbf{A} : $\text{tr}(\mathbf{A}) = \sum_i a_{ii}$

Abbildung 3.3: Moments of a vector

special case $d = 1$: $\underline{X} \rightarrow X \in \mathbb{R}$

$\mu \rightarrow \mu \in \mathbb{R}$

$\underline{\underline{C}} \rightarrow$ variance of $X = \text{Var}(X) = \delta^2 = E[(X - \mu)^2] = \dots = E(X^2) - \mu^2$

$\underline{\underline{\delta}} = \sqrt{\text{Var}(X)}$: standard deviation

For any function $g(\underline{X})$ of \underline{X} : $E[g(\underline{x})] = \int g(x) \cdot p(\underline{x}) d\underline{x} = (\text{d.v.}) \sum_i g(\underline{x}_i) \cdot P(\underline{x}_i)$

Multivariate normal (Gaussian) distribution

PDF

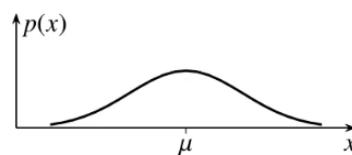
$$\underline{X} \in \mathbb{R}^d \sim N(\underline{\mu}, \mathbf{C})$$

$$p(\underline{x}) = \frac{1}{(2\pi)^{d/2} |\mathbf{C}|^{1/2}} e^{-\frac{1}{2}(\underline{x}-\underline{\mu})^T \mathbf{C}^{-1} (\underline{x}-\underline{\mu})}$$

$$\ln(p(\underline{x})) = -\frac{d}{2} \ln(2\pi) - \frac{1}{2} \ln(|\mathbf{C}|) - \frac{1}{2}(\underline{x}-\underline{\mu})^T \mathbf{C}^{-1} (\underline{x}-\underline{\mu})$$

$N(\underline{0}, \mathbf{I})$ is called the **standard normal distribution**.

1D-Visualization



$$\mathbf{I} = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & 0 & & 0 \end{bmatrix}$$

identity matrix

Moments

$$E(\underline{X}) = \underline{\mu}, \quad \text{Cov}(\underline{X}) = \mathbf{C}$$

Abbildung 3.4: Multivariate normal (Gaussian) distribution

one-hot coding : Only one bit is 1 e.g. 0100 one cold coding is the inverse

Coding for class label, random vector y of length c so the identity matrix with dimension c is used for class labels

Reformulation of the PMF (categorical distribution) by one-hot coding of the classes

$\underline{x} = [x_i] \in \{\underline{e}_1, \underline{e}_2, \dots, \underline{e}_c\}$, i.e. all $x_i = 0$ except for one single element equal to 1

$$\text{PMF: } P(\underline{X} = \underline{x}) = P(\underline{x}) = \begin{cases} P_i & \text{if } \underline{x} = \underline{e}_1 \text{ or } x_1 = 1 \\ \dots & \\ P_c & \text{if } \underline{x} = \underline{e}_c \text{ or } x_c = 1 \end{cases} = p_1^{x_1} \cdot p_2^{x_2} \cdot \dots \cdot p_c^{x_c} = \prod_{i=1}^c p_i^{x_i}$$

$$\ln(P(\underline{x}) = \sum_{i=1}^c x_i \cdot \ln(p_i) = [x_1, \dots, x_c] \cdot \begin{bmatrix} \ln(P_1) \\ \dots \\ \ln(P_c) \end{bmatrix} = \underline{x}^T \cdot \ln(\underline{P})$$

\ln function applied element wise

3.3. Chapter 3.3 - Multiple random vectors

- 3-16 Table for distributions

- product rule for probability $p(\underline{x}, \underline{y}) = p(\underline{x}) \cdot p(\underline{y})$

- Bayes rule $p(\underline{y}|\underline{x}) = p(\underline{y}|\underline{x}) \cdot \frac{p(\underline{x})}{p(\underline{y})}$

- Independent and identically distributed

\underline{x} and \underline{y} are independent if:

$$p(\underline{x}, \underline{y}) = p(\underline{x}) \cdot p(\underline{y}) \leftrightarrow p(\underline{x}|\underline{y}) = p(\underline{x})$$

$\underline{x}_1, \dots, \underline{x}_N$ are independent and identically distributed (i.i.d)

$$P(\underline{x}_1, \dots, \underline{x}_N) = \prod_{i=1}^N p_i(\underline{x}_i), \underline{X}_i \sim p_i(\underline{x}_i)$$

$$p_i(\underline{x}_i) = p(\underline{x}_i) \rightarrow p(\underline{x}_1, \dots, \underline{x}_N) = \prod_{i=1}^N p(\underline{x}_i)$$

3.3.1. Chapter 3.2.3 - Kernel based density estimation

PDF: $p(\underline{x})$ of $\underline{X} \in \mathbf{R}^d$ unknown, only i.i.d samples $\underline{x}(n), 1 \leq n \leq N$

kernel-based estimate $\hat{p}(\underline{x})$ of $p(\underline{x})$ from $\underline{x}(n)$

kernel function $k(\underline{x})$, like a PDF

$$1. k(\underline{x}) \geq 0 \forall \underline{x}$$

$$2. \int k(\underline{x}) d\underline{x} = 1$$

$$\hat{p}(\underline{x}) = \frac{1}{N} \sum_{n=1}^N k(\underline{x} - \underline{x}(n))$$

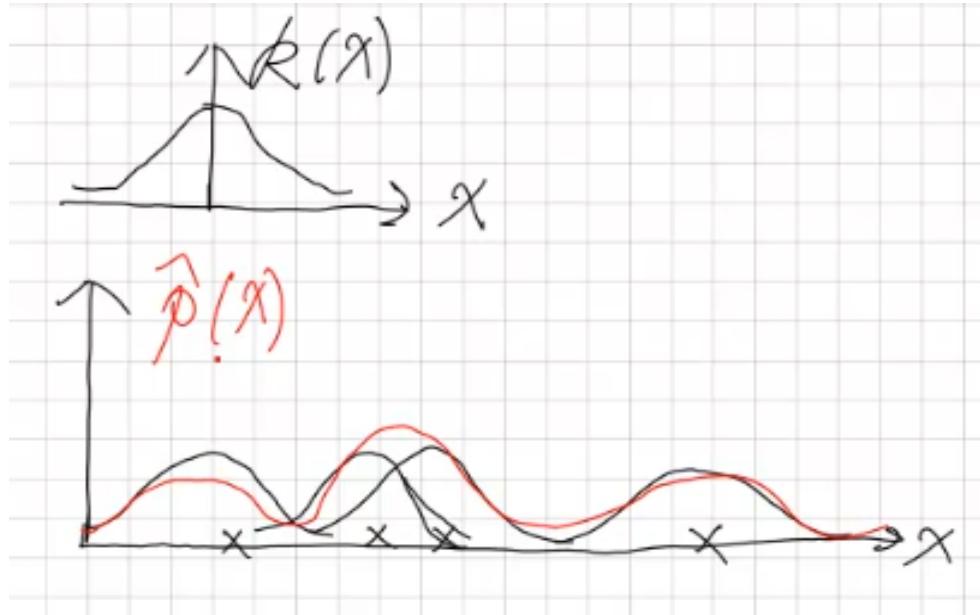


Abbildung 3.5: Kernel function

Smooth Gaussian Kernel:

$$N(\underline{0}, \underline{\underline{I}}) : k(\underline{x}) = \frac{1}{2\pi^{\frac{d}{2}}} \cdot e^{-\frac{1}{2}\|\underline{x}\|^2}$$

Dirac Kernel

$k(\underline{x}) = \delta(\underline{x})$: Dirac function

$$\delta(\underline{x}) = \begin{cases} \infty, \underline{x} = \underline{0} \\ 0, \underline{x} \neq \underline{0} \end{cases}$$

$$\int \delta(\underline{x}) d\underline{x} = 1$$

sampling property : $\int \delta(\underline{x} - \underline{x}_0) f(\underline{x}) d\underline{x} = f(\underline{x}_0)$

empirical distribution

$$\hat{p}(\underline{x}) \cdot \frac{1}{N} \sum_{n=1}^N \delta(\underline{x} - \underline{x}(n))$$

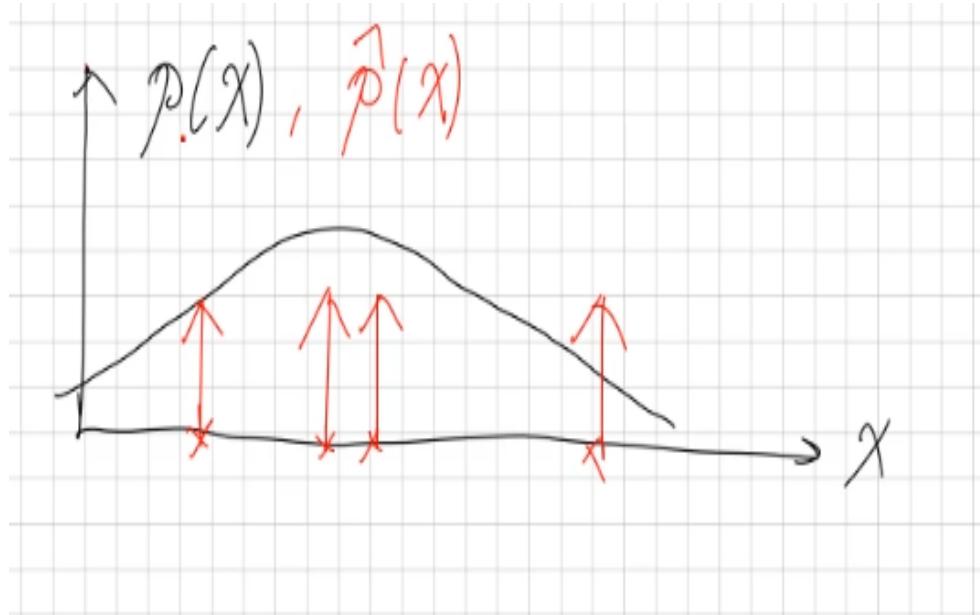


Abbildung 3.6: Estimated PDF function

3.4. Chapter 3.4 Kullback-Leibler divergence and cross entropy

Dissimilarity measure between 2 distributions:

Case A: continuous-valued random variables: PDF

$\underline{X} \sim p(\underline{x})$: true statistical distribution of \underline{X}

$q(\underline{x})$: approximation for $p(\underline{x})$, e.g. by DNN

KL divergence (KLD) between p and q:

$$D_{KL}(p||q) = \int p(\underline{x}) \cdot \ln\left(\frac{p(\underline{x})}{q(\underline{x})}\right) d\underline{x}$$

$$= E_{\underline{X} \sim p} \left[\ln\left(\frac{p(\underline{X})}{q(\underline{X})}\right) \right]$$

expectation over $p(\underline{x})$

DKL is real valued scalar positive or negative or 0

Case B: discrete-valued random vector: PMF

$\underline{X} \in \{\underline{x}_1, \dots, \underline{x}_c\} \sim$: true PMF of $\underline{X} \sim Q(\underline{x})$: approximation for $P(\underline{x})$

$$D_{KL}(P||Q) = \sum_{i=1}^c P(\underline{x}_i) \cdot \ln\left(\frac{P(\underline{x}_i)}{Q(\underline{x}_i)}\right) = E_{\underline{X} \sim P} \left[\ln\left(\frac{P(\underline{X})}{Q(\underline{X})}\right) \right]$$

Properties of the KL divergence: P1) Nonnegative $D_{KL}(P||Q) \geq 0 \forall p, q$

P2) Equality $D_{KL}(P||Q) = 0$ iff(if and only if) $p(\underline{x}) = q(\underline{x})$

proof for "sufficient": $\ln\left(\frac{p(\underline{x})}{q(\underline{x})}\right) = 0 \forall \underline{x}$

P1 and P2: $D_{KL}(p||1)$ is a suitable metric for approximation p by q

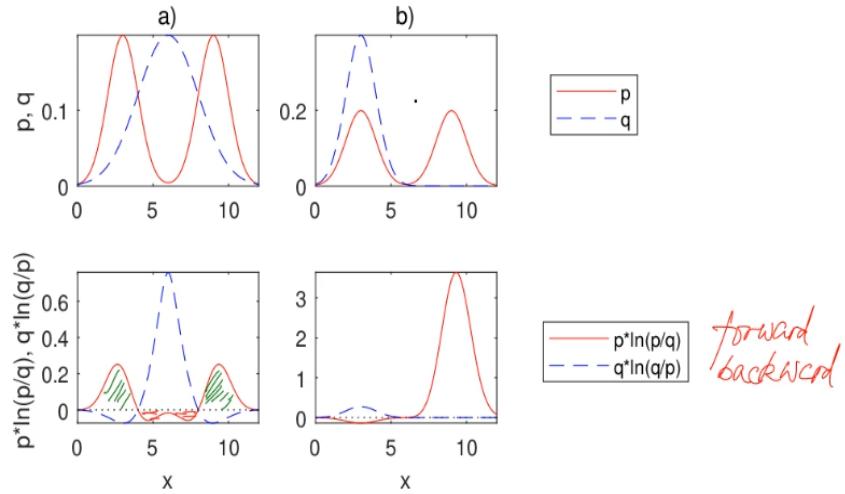
P3 Asymmetry

$$D_{KL}(p||q) = E_{\underline{X} \sim p} \left[\ln\left(\frac{p(\underline{x})}{q(\underline{x})}\right) \right] \neq D_{KL}(q||p) = E_{\underline{X} \sim q} \left[\ln\left(\frac{q(\underline{x})}{p(\underline{x})}\right) \right]$$

forward KLD

backward KLD

D_{KL} is not a true distance measure with $D(\underline{x}, \underline{y}) = D(\underline{y}, \underline{x})$



- When minimizing $D_{KL}(p||q)$, a) is better than b) because $q(x)$ is broad.
- When minimizing $D_{KL}(q||p)$, b) is better than a) because $q(x)$ is narrow.

Abbildung 3.7: Forward vs. Backward KL divergence

3.4.1. E3.5 KLD between normal and Laplace distribution

$$p(x) = \sim N(0, \sigma^2), p(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{x^2}{2\sigma^2}}$$

$$q(x) \sim \text{Laplace}: (0, b) q(x) = \frac{1}{2b} e^{-\frac{|x|}{b}}$$

Task: choose b to best approximate p by q.

$$\frac{p(x)}{q(x)} = \sqrt{\frac{2}{\pi}} \cdot \frac{b}{\sigma} \cdot \exp\left(-\frac{x^2}{2\sigma^2} + \frac{|x|}{b}\right)$$

$$D_{KL}(p||q) = E_{\underline{X} \text{ simp}} = \ln\left(\frac{p(x)}{q(x)}\right) = \ln\left(\sqrt{\frac{2}{\pi}} \cdot \frac{b}{\sigma}\right) + E_{X \sim p}\left(\left(-\frac{x^2}{2\sigma^2} + \frac{|x|}{b}\right)\right)$$

$$E_{\underline{X} \text{ simp}} = \sigma^2$$

$$E_{\underline{X} \text{ simp}}(|x|) = \int_{-\infty}^{\infty} |x| \cdot \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right) dx = 2 \int_0^{\infty} |x| \cdot \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right) dx = \sqrt{\frac{2}{\pi}} \cdot \sigma$$

$$\text{Let } \alpha = \frac{\sigma}{b}, D_{KL}(p||q) = \dots = \sqrt{\frac{2}{\pi}} \cdot \alpha - \ln(\alpha) + \ln\left(\sqrt{\frac{2}{\pi}}\right) - \frac{1}{2} \frac{dD_{KL}(p||q)}{d\alpha} = \sqrt{\frac{2}{\pi}} - \frac{1}{\alpha} = 0 \rightarrow \alpha = \sqrt{\frac{2}{\pi}}, \text{ i.e. } b \approx 0,86$$

$$D_{KL,min}(p||q) = D_{KL}(p||q)|\alpha = \sqrt{\frac{2}{\pi}} = \dots = \frac{1}{2} - \ln\left(\frac{\pi}{2}\right) \approx 0,048$$

• Probable exam question calculate this for 2 distributions

P4) Additive :

$\underline{X} = (\underline{x}_1, \underline{x}_2)$, \underline{x}_1 and \underline{x}_2 are independent, i.e.

$$p(\underline{x}) = p_1(\underline{x}_1) \cdot p_2(\underline{x}_2), q(\underline{x}) = q_1(\underline{x}_1) \cdot q_2(\underline{x}_2)$$

$$\text{Then: } D_{KL}(p||q) = D_{KL}(p_1||q_1) + D_{KL}(p_2||q_2)$$

P5) Relation to cross entropy :

Definiton of entropy 3-24 probability always greater than 0 but smaller than 1

$$D_{KL}(p||q) = \int p \ln\left(\frac{p}{q}\right) dx = \int p \ln(p) dx - \int p \ln(q) dx = -H(p) + H(p, q) \text{ or}$$

$$\text{cross entropy: } H(p, q) = D_{KL}(p||q) + H(p) \geq H(p) \geq 0$$

For a given (fixed) $p(\underline{x})$: $H(p)$ fixed

Hence: $\min D_{KL}(p||q) \leftrightarrow \min H(p, q)$

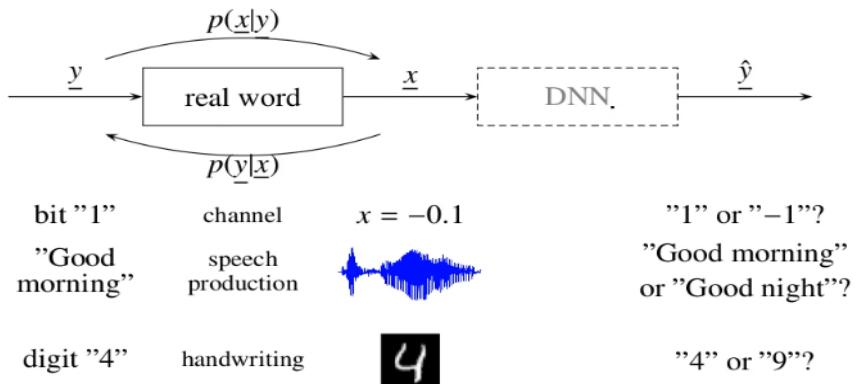
Not the case for backward KLD $D_{KL}(q||p)$!

Minimizing is not the same anymore because then it is $H(q)$ and that's what we are trying to optimize

3.5. Chapter 3.5 Probabilistic framework

valid for both SP and ML

valid for both regression problem and classification problem



- \underline{y} : **latent variable**, hidden, not directly measurable, quantity of interest
- \underline{x} : **observed variable**, measurement, input for DNN
- $\hat{\underline{y}}$: output of DNN as estimate for \underline{y}
- real world: describes how \underline{x} is generated from \underline{y}
- DNN: describes how to estimate \underline{y} from \underline{x}

Abbildung 3.8: Probabilistic framework of supervised learning

Both \underline{y} and \underline{x} are modeled as random variables. They are described by the joint **data generating distribution**

$$p(\underline{x}, \underline{y}) = p(\underline{x}|\underline{y})p(\underline{y}) = p(\underline{y}|\underline{x})p(\underline{x}).$$

$p(\underline{y})$ prior PDF of \underline{y} or **prior**, available before any measurement of \underline{x}

$p(\underline{x}|\underline{y})$ **likelihood**. It describes the real word, the generation of \underline{x} from \underline{y} .

It is a kind of channel-sensor model, e.g.

- bit: communication channel + receiver
- speech: speech production system + microphone
- digit: handwriting + camera

$p(\underline{x})$ prior PDF of \underline{x} , also called **evidence**

$p(\underline{y}|\underline{x})$ posterior PDF of \underline{y} after a measurement \underline{x} or **posterior**

Abbildung 3.9: The data generating distribution

Bayes Rule:

Bayes rule:

$$p(\underline{y}|\underline{x}) = p(\underline{x}|\underline{y}) \cdot \frac{p(\underline{y})}{p(\underline{x})}$$

posterior likelihood

prior
evidence

i.e. $p(\underline{y}|\underline{x})$ contains both $p(\underline{x}|\underline{y})$ and $p(\underline{y})$

Training in ML: calculate/model $p(\underline{y}|\underline{x}) \forall \underline{x}, \underline{y}$ from D_{train}

Inference ... : Draw conclusion about \underline{y} for a given \underline{x} based on $p(\underline{y}|\underline{x})$

Abbildung 3.10: Bayes Rule in DL

a) **Maximum a posterior (MAP) inference:**

$$\max_{\underline{y}} p(\underline{y}|\underline{x}) = p(\underline{x}|\underline{y}) \frac{p(\underline{y})}{p(\underline{x})}$$

b) **Minimum Bayesian risk (MBR) inference:**

$$\min_{\hat{\underline{y}}} E_{\underline{X}, \underline{Y}}[l(\underline{Y}, \hat{\underline{y}}(\underline{X}))] = \int l(\underline{y}, \hat{\underline{y}}(\underline{x})) p(\underline{x}, \underline{y}) d\underline{x} d\underline{y}$$

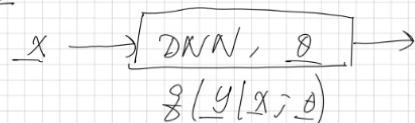
see course DPR and SASP for more details.

Abbildung 3.11: Bayes decision theorem

Supervised learning:

* $p(\underline{x}|\underline{y})$, $p(\underline{y})$ unknown $\rightarrow p(\underline{y}|\underline{x})$ unknown

* approximate $p(\underline{y}|\underline{x})$ by a parametric posterior model
 $g(\underline{y}|\underline{x}; \underline{\theta})$, given by a DNN with parameter vector $\underline{\theta}$



* function $g(\cdot)$ known \leftarrow DNN architecture

* $\underline{\theta}$ unknown \leftarrow coefficients

* learning $\underline{\theta}$ from D_{train}

Abbildung 3.12: Supervised learning

Learning Criterion:

Learning criterion:

$$\min_{\theta} D_{KL}(p(x, y) \parallel q(x, y; \theta)) \quad \xleftarrow{p(x, y) \text{ fixed}}$$

$$\min_{\theta} H(p, q)$$

Since $q(x, y; \theta) = q(y|x; \theta) \cdot q(x)$,

$$CE \quad H(p(x, y), q(x, y; \theta)) = - \int p(x, y) \ln q(x, y; \theta) dx dy$$

$$= - \int p(x, y) \cdot \ln q(y|x; \theta) dx dy$$

$$- \underbrace{\int \dots \ln q(x) dx dy}_{\text{independent of } \theta, \text{ const}}$$

Abbildung 3.13: Calc part1

$$= \int p(x, y) \cdot \underbrace{[-\ln q(y|x; \theta)]}_{\text{loss}} dx dy + \text{const.}$$

average loss, Bayesian risk, see DPR

In practice: $p(x, y)$ replaced by the empirical distib.

$$\hat{p}(x, y) = \frac{1}{N} \sum_{n=1}^N \delta(x - x(n), y - y(n))$$

3.2.3: sampling property of $\delta(\cdot)$:

$$\min_{\theta} H(p, q) = \text{const} + \frac{1}{N} \sum_{n=1}^N \underbrace{[-\ln q(y(n)|x(n); \theta)]}_{\text{loss } L(x(n), y(n); \theta)}$$

cost function $L(\theta)$

Abbildung 3.14: Calc part2

3.5.1. Role of a NN

1. approximate true posterior $p(y|x)$ by $q(y|x; \Theta)$
2. learn Θ from D_{train}

Lecture 4: Dense Neural Networks

Date: 05/05/2020

Lecturer: David Silver

By: Nithish Moudhgalya

A general model for $q(y|\underline{x}; \Theta)$

*) can learn any linear or nonlinear mapping

*) suitable for both regression and classification problems

artificial NN: mimic biological NN (brain)

4.0.1. 4.1 Fully connected neural networks - Neuron

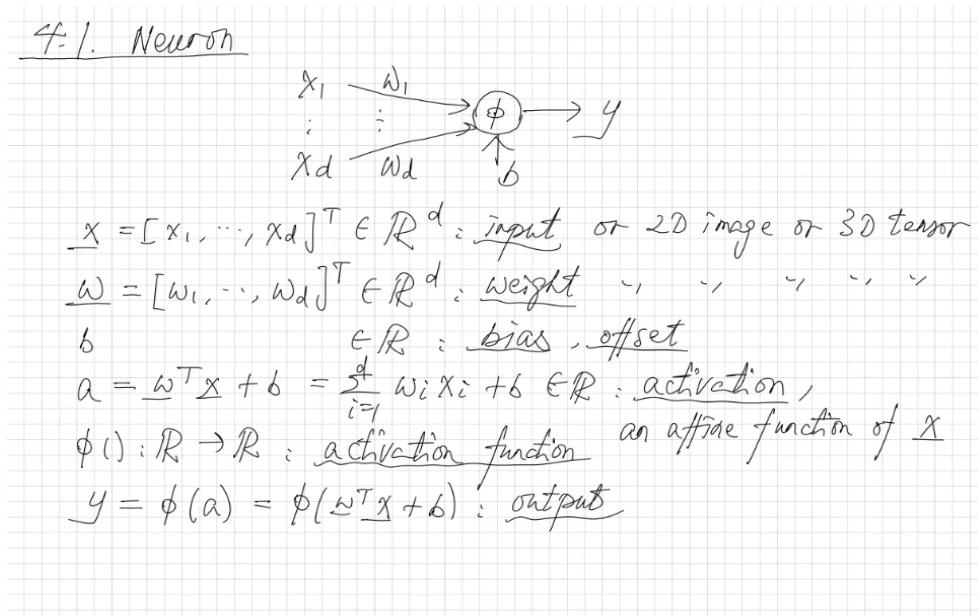


Abbildung 4.1: Neuron

4.0.2. Chapter 4.2 Layer of Neurons

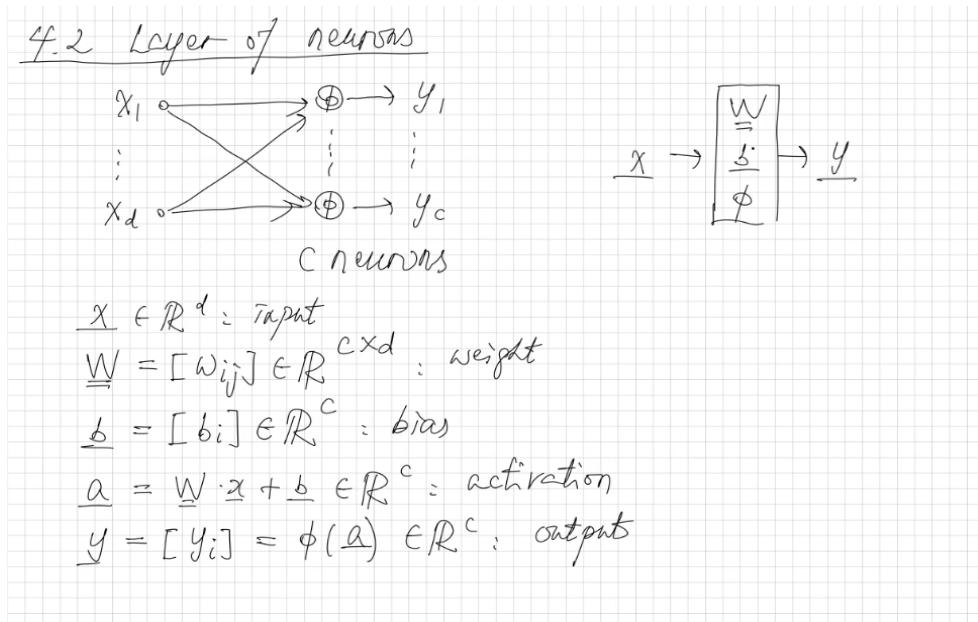


Abbildung 4.2: Layer of Neurons

2 meanings of $\phi()$:

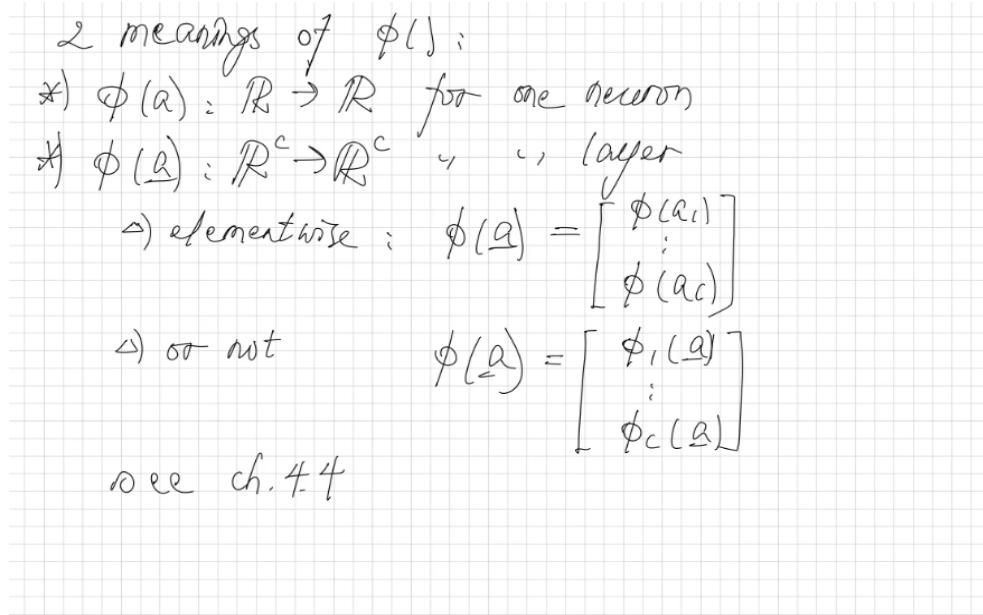


Abbildung 4.3: Meanings of phi

Comments :

- no interconnections between neurons in the same layer
- dense layer, fully connected layer: • each input x_j connected to each neuron i
 $\rightarrow c \cdot d$ weights and w_{ij} an c biases b_i , $1 \leq i \leq c, 1 \leq j \leq d$,
 $\rightarrow c \cdot (d + 1)$ parameters

4.1. 4.3 Feedforward neural network

A cascade of dense layers

Layer $1 \leq l \leq L$:

M_l - number of the input neurons

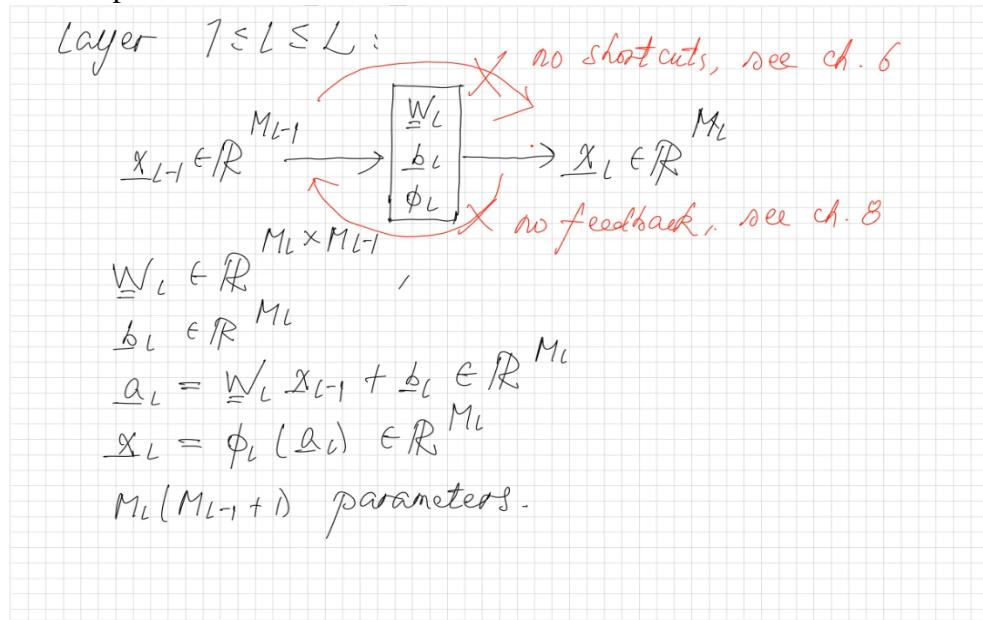


Abbildung 4.4: Feedforward multilayer neural network

Network:

$$x_L = f(x_0; \underline{\theta}) : \mathbb{R}^{M_0} \rightarrow \mathbb{R}^{M_L}$$

* parameter vector $\underline{\theta} = \begin{bmatrix} \text{vec}(W_1) \\ b_1 \\ \vdots \\ \text{vec}(W_L) \\ b_L \end{bmatrix} \in \mathbb{R}^{N_p}$

learned from Train

Number of parameters: $N_p = \sum_{l=1}^L M_l(M_{l-1} + 1)$

~,~, multiplications: $N_x = \prod_{l=1}^L M_l M_{l-1}$

* $L, \{M_1, \dots, M_L\}$ } hyperparameters chosen by you

* $\{\phi_1, \dots, \phi_L\}$ }

Abbildung 4.5: Network Parameters

Lecture 4: Dense Neural Networks

Date: 15/05/2020

Lecturer: David Silver

By: Nithish Moudhgalya

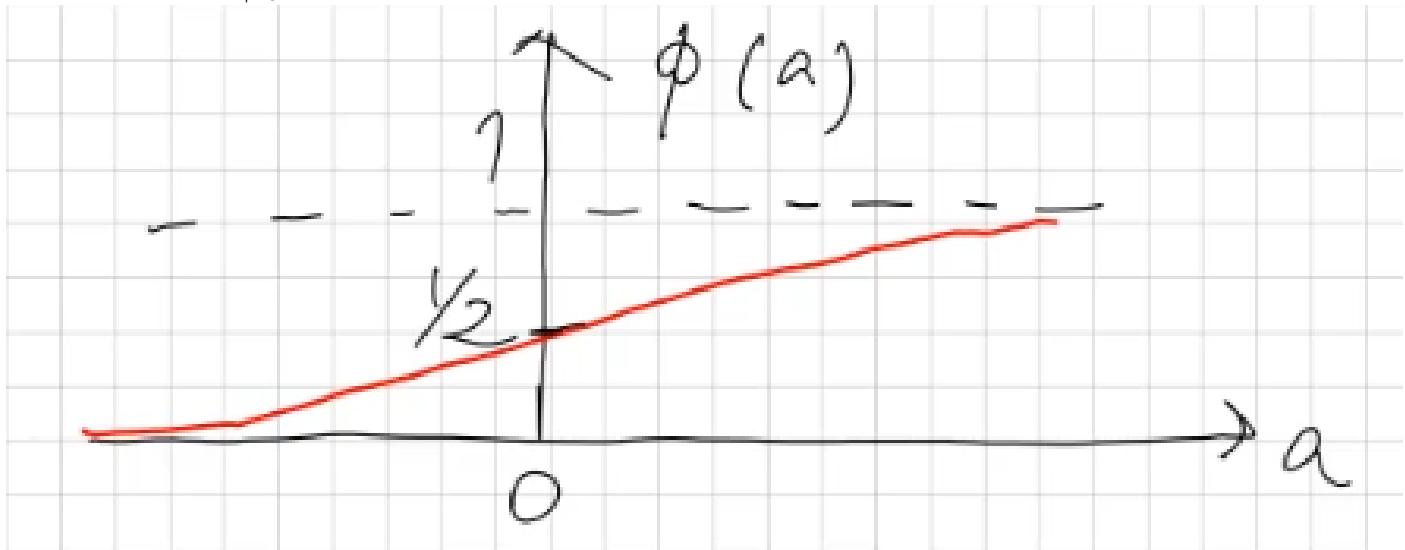
5.1. Activation function

Mild requirements on $\phi()$:

- nonlinear in general \rightarrow fundamental
- smooth, differentiable \rightarrow for training
- simple calculation \rightarrow low complexity
- Slides 4-6; 4-7 activation function types

5.1.1. Sigmoid activation function

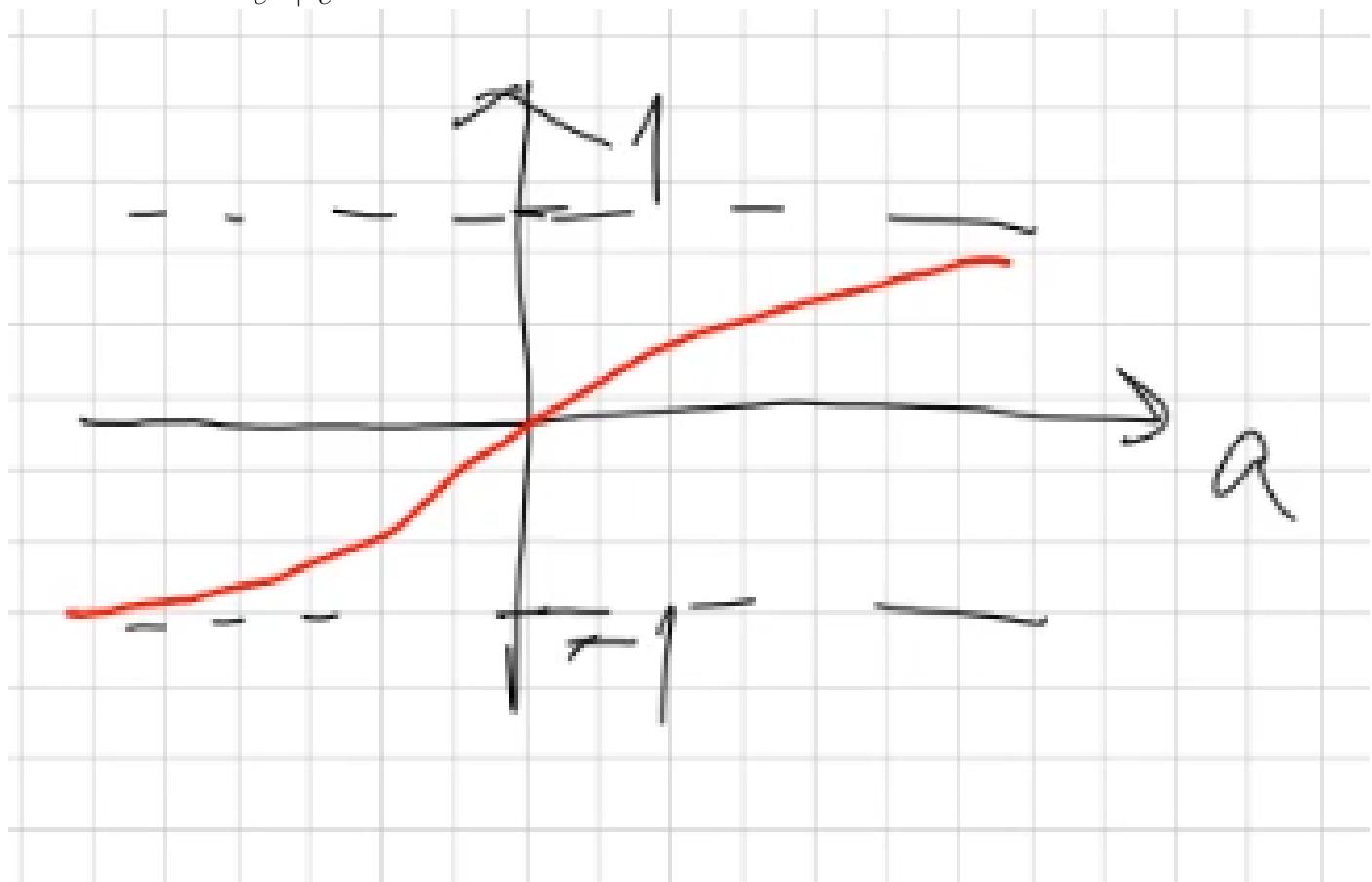
$$\phi(a) = \sigma(a) = \frac{1}{1 + e^{-a}}$$



- $0 < \phi < 1$ $\hat{=}$ prob.
- symmetry: $\phi(-a) = 1 - \phi(a)$
- derivative: $\frac{d\phi(a)}{da} = \dots = \frac{e^{-a}}{(1 + e^{-a})^2} = \phi(a) \cdot \phi(-a) = \phi(a)(1 - \phi(a)) \in (0, 1)$
easy calculative
- widely used in conventional NN (shallow)

5.1.2. hyperbolic tangent activation function

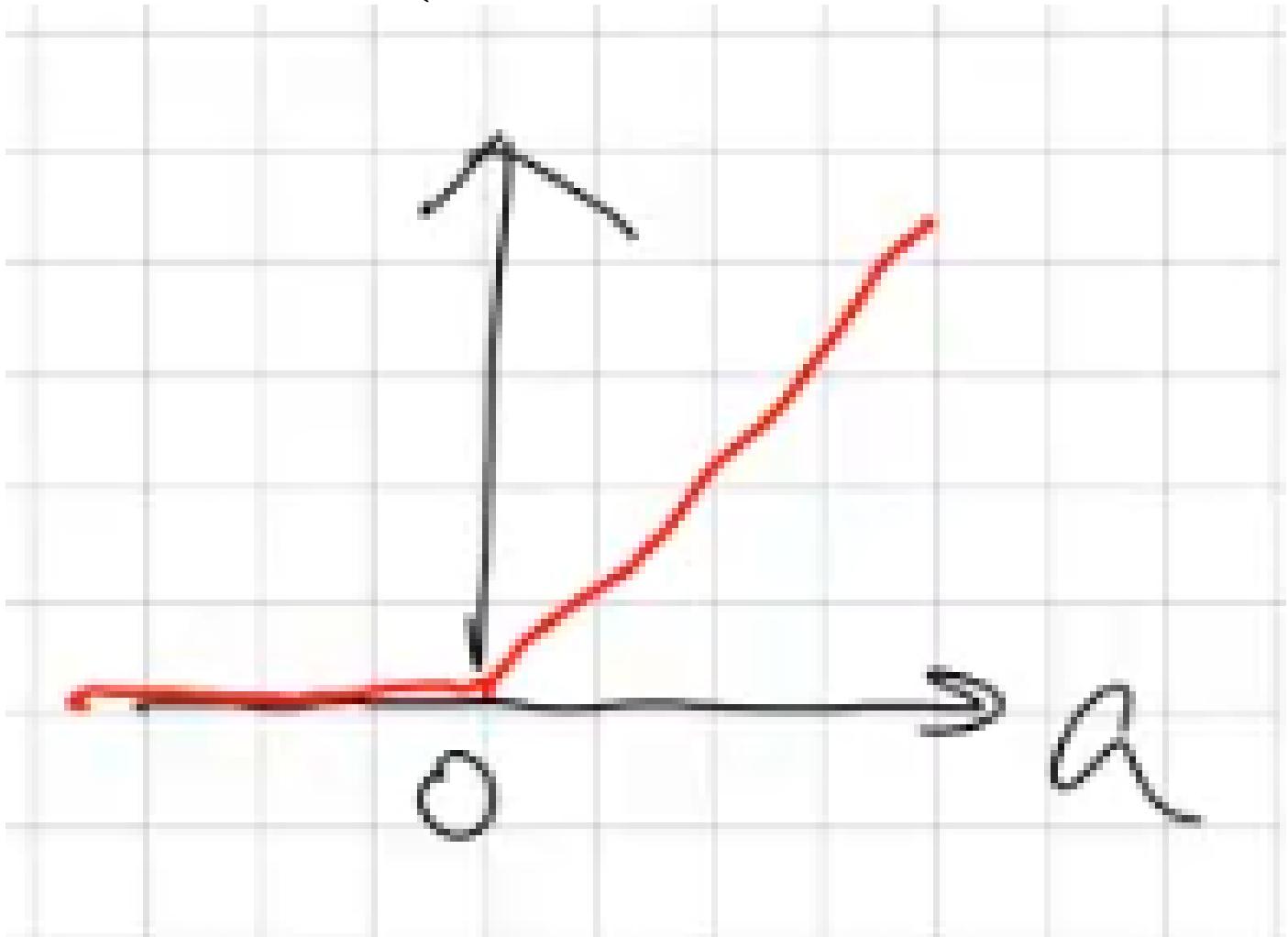
$$\phi(a) = \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$



like sigmoid but another output range

5.1.3. rectifier linear unit(ReLU)

$$\phi(a) = \text{ReLU}(a) = \max(a, 0) = \begin{cases} a & a \geq 0 \\ 0 & a < 0 \end{cases}$$



- ≡ diode

- simple calculation

- $\frac{d\phi}{da} = \begin{cases} 1 & a > 0 \\ 0 & a < 0 \end{cases} = u(a), u(0) = 0$ typically used

- most popular in DNN

- Details on 4-7

5.1.4. Softmax activation function(classification problem)

$$\phi(\underline{a} : \underline{a} = [a_i] \in \Re^c \rightarrow \Re^c)$$

$$\phi(\underline{a}) = \text{softmax}(\underline{a}) = \begin{bmatrix} \phi_1(\underline{a}) \\ \vdots \\ \phi_c(\underline{a}) \end{bmatrix}$$

$$\phi(\underline{a}) = \frac{a^{a_i}}{\sum_{j=1}^c e^{a_j}}, \in (0, 1), \sum_{i=1}^c \phi_i(\underline{a}) = 1$$

- maps $\underline{a} \in \Re^c$ to a categorical PMF with c classes

- a_i large $\rightarrow \phi_i(\underline{a})$ close to 1

- a_i small $\rightarrow \phi_i(\underline{a})$ close to 0

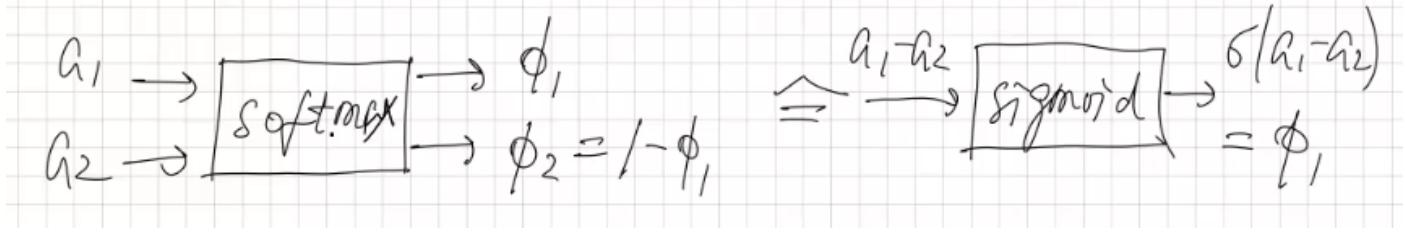
- used in the output layer for classification problems

5.2. Special case c=2, binary classification problem

$$\phi_1(\underline{a}) = \frac{e^{a_1}}{e^{a_1} + e^{a_2}} = \frac{1}{1 + e^{-(a_1 - a_2)}} = \sigma(a_1 - a_2)$$

$$\phi_2(\underline{a}) = \frac{e^{a_2}}{e^{a_1} + e^{a_2}} = 1 - \phi_1(\underline{a}) = \sigma(a_2 - a_1)$$

i.e. softmax



one sigmoid output is sufficient for binary classification instead of 2 output softmax!

Derivative of softmax:

$$\frac{\partial \phi_i(\underline{a})}{\partial a_j} = \dots = \begin{cases} \phi : i(\underline{a}) \cdot (1 - \phi_i(\underline{a})) & i = j \\ -\phi_i(\underline{a}) \cdot \phi_j(\underline{a}) & i \neq j \end{cases} \quad \text{•4-9 for details on usage}$$

5.3. Chapter 4.5 Universal approximation

4.5 Universal approximation

Universal approximation theorem $\stackrel{4-10}{\cong}$ *existence of a solution*

The **universal approximation theorem** states that a feedforward neural network with a linear output layer ($\phi_L(\underline{a}) = \underline{a}$) and

- at least one hidden layer with
- a nonlinear activation function

can approximate any continuous (nonlinear) function $y(\underline{x}_0)$ (on compact input sets) to arbitrary accuracy.

Comments:

- arbitrary accuracy: with an increasing number of hidden neurons.
- valid for a wide range of nonlinear activation functions, but excluding polynomials.
- minimum requirement for universal approximation: $\mathbf{W}_2\phi_1(\mathbf{W}_1\underline{x}_0 + \underline{b}_1) + \underline{b}_2$.

For learning \mathbf{W}_l and \underline{b}_l from D_{train} :

- deep networks are better than shallow ones,
- some activation functions are better than others.

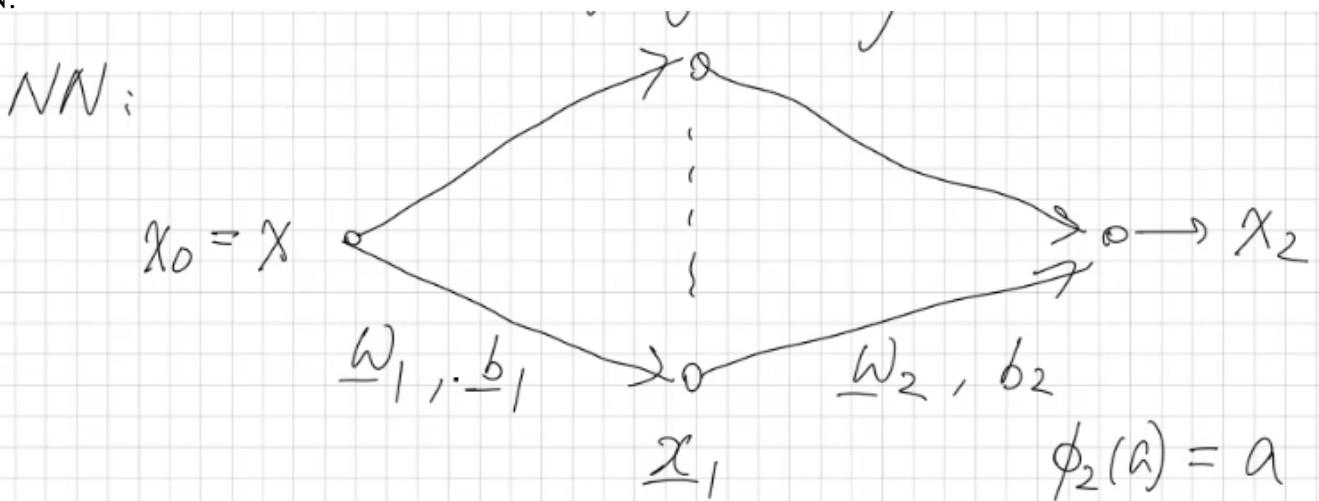
\cong how to find a good solution.

+

5.3.1. E4.3 Regression with 1 hidden layer

True function: $f_0(x)$

Given: $x(n)$ and noisy function $y(n) = f(x(n)) + z(n)$, $1 \leq n \leq N$, $z(n)$ is the noise
NN:



M_1 hidden neurons

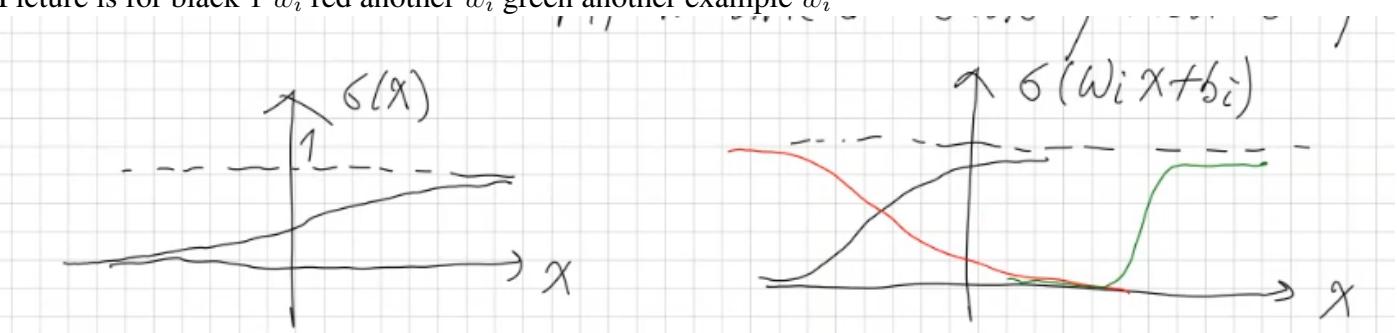
Sigmoid $\phi_i(a) = \sigma(a)$

i.e. $x_2 = f(x, \theta) = w_2^T \sigma(w_1 x + b_1)$, column times row times scalar
 $= \sum_{i=1}^{N_1} w_{2,i} \cdot \underbrace{\sigma(w_{1,i} x + b_{1,i})}_{M_1 \text{ nonlinear basis functions of } x} + b_2$

$\sigma(w_i x + b_i) \sigma(w_i(x + \frac{b_i}{w_i}))$

new center at $\frac{-b_i}{w_i}$ and new slope value w_i

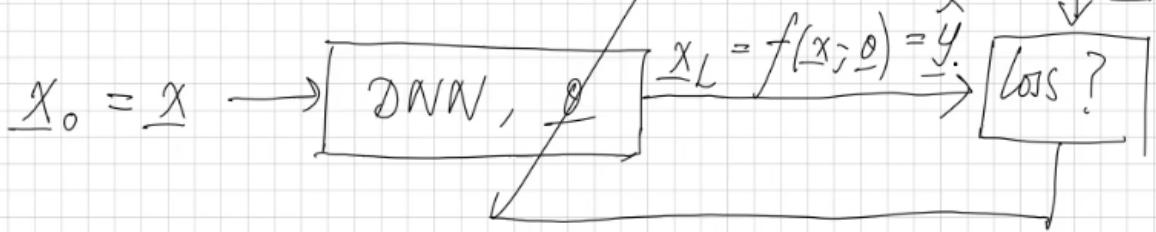
Picture is for black 1 w_i red another w_i green another example w_i



5.4. Chapter 4.4 - 4.6 Loss and cost function

Review chapter 3.4

Ch. 3-4:



$$\min_{\theta} L(\underline{\theta}) = \frac{1}{N} = \sum_{n=1}^N (l(\underline{x}(n), y(n)) \underline{\theta}) \text{ cost function for } d_{train}$$

$$l(\underline{x}, \underline{y} | \underline{\theta}) = -\ln(q(\underline{y} | \underline{x}, \underline{\theta})) \text{ loss for one pair } (\underline{x}, \underline{y})$$

$$q() \leftrightarrow \text{DNN} ???$$

4.6.1 Regression Problem

$\underline{x} \in \Re^d$: random input

$\underline{y} \in \Re^c$ desired random output

Assumption: DNN estimates the mean of \underline{y} , i.e.

$$\underline{y} = f(\underline{x}, \underline{\theta}) + \underline{z}, \underline{z} : \text{noise}$$

f is calculated by DNN

case A: $\underline{z} \sim N(0, \sigma^2 \underline{I})$, white Gaussian noise

$$\underline{y} \sim N(f(\underline{x}, \underline{\theta}), \sigma^2 \underline{I})$$

$$q(\underline{y} | \underline{x}, \underline{\theta}) = \frac{1}{(2\pi\sigma^2)^{c/2}} \exp\left(-\frac{1}{2\sigma^2} \|\underline{y} - f(\underline{x}, \underline{\theta})\|^2\right)$$

$$l(\underline{x}, \underline{y}, \underline{\theta}) = -\ln(q(\underline{y} | \underline{x}, \underline{\theta})) = \text{const.} + \frac{1}{\sigma^2} \|\underline{y} - f(\underline{x}, \underline{\theta})\|^2$$

$$L(\underline{\theta}) = \frac{1}{N} \sum_{n=1}^N \|\underline{y}(n) - f(\underline{x}, \underline{\theta})\|^2$$

$\rightarrow l_2 - \text{loss}$, mean square error (MSE) loss, least squares (LS) method

$\min L(\underline{\theta})$ is a parameter estimation problem

case B:

$\underline{z} \sim N(0, \underline{\underline{C}})$, colored Gaussian noise

$$L(\underline{\theta}) = \frac{1}{N} \sum_{n=1}^N [\underline{y} - (f(\underline{x}(n), \underline{\theta}))]^T \cdot \underline{\underline{C}}^{-1} \cdot [\underline{y} - (f(\underline{x}), \underline{\theta})], \text{ weighted MSE loss}$$

Rarely used in real life applications:

- how to know $\underline{\underline{C}}$

- $\underline{\underline{C}}^{-1}$ expensive for large $\underline{\underline{C}}$

4.6.2 Classification

\underline{x} in \Re^d : input

$$\underline{y} \in \{\underline{e}_1, \underline{e}_2, \dots, \underline{e}_c\}$$

: class label for \underline{x} in one-hot coding

Let $p_i = P(y = \underline{e}_i | \underline{x})$: true posterior probability

ch: 3.2 : $P(\underline{y} | \underline{x}) = \prod_{i=1}^c p_i^{y_i}$ true PMF

But p_i unknown

DNN: • output $f(\underline{y}, \underline{\theta}) = [f_i(\underline{x}; \underline{\theta})] \in \Re^c$ an estimate for $[p_i]$

• i. e. $P(\underline{y}|\underline{x})$ approximated $Q(\underline{y}|\underline{x}; \underline{\theta}) = \prod_{i=1}^c f_i(\underline{x}; \underline{\theta})^{y_i}$

in order to ensure :

• $0 < f_i(\underline{y}; \underline{\theta}) < 1$

• $\sum_{i=1}^c f_i(\underline{x}; \underline{\theta}) = 1$,

softmax is used in the output layer :

$\underline{x}_L = f(\underline{x}; \underline{\theta}) = \phi_L(\underline{a}_L) = \text{softmax}(\underline{a}_L)$, see ch.4.4

→ Loss $l(\underline{x}, \underline{y}; \underline{\theta}) = -\ln(Q(\underline{y}|\underline{x}; \underline{\theta})) = \sum_{i=1}^c y_i \ln(f_i(\underline{y}; \underline{\theta})) = -\underline{y}^T \ln(f(\underline{x}; \underline{\theta})) \geq 0$

categorical cross entropy(CE) loss

Special case : Binary classification , $c = 2$

ch.4.4: softmax, $c = 2 \Leftarrow \text{sigmoid}$

i.e. one output neuron with sigmoid activation function $f(\underline{x}; \underline{\theta}) = \sigma(a_L)$ is sufficient

Let $y_1 = y, y_2 = 1 - y; f_1 = f; f_2 = 1 - f$

→ $l(\underline{x}, \underline{y}; \underline{\theta}) = -y \ln(f(\underline{x}; \underline{\theta}) + (1 - y)) \cdot \ln(1 - f(\underline{x}; \underline{\theta}))$, binary CE loss

True probabilistic way toward cost functions

The probabilistic way toward the cost functions

- $p(\underline{x}, \underline{y})$: true but unknown data generating distribution, application specific
- $D_{\text{train}} = \{\underline{x}(n), \underline{y}(n), 1 \leq n \leq N\}$: training set, i.i.d. samples of $p(\underline{x}, \underline{y})$
- $p(\underline{y}|\underline{x})$: true posterior describing the desired inference $\underline{x} \rightarrow \underline{y}$
- $q(\underline{y}|\underline{x}; \underline{\theta})$: a parametric model (DNN) to approximate $p(\underline{y}|\underline{x})$

min. forward KL divergence $D_{\text{KL}}(p(\underline{x}, \underline{y}) \| q(\underline{x}, \underline{y}; \underline{\theta}))$

↓ ch. 3.3: p fixed

min. cross entropy $H(p, q) = -\mathbb{E}_{\underline{X}, \underline{Y} \sim p(\underline{x}, \underline{y})} \ln(q(\underline{Y}|\underline{X}; \underline{\theta}))$

↓ ch. 3.4: use empirical distribution $\hat{p}(\underline{x}, \underline{y})$

min. cross entropy $H(\hat{p}, q) = -\mathbb{E}_{\underline{X}, \underline{Y} \sim \hat{p}(\underline{x}, \underline{y})} \ln(q(\underline{Y}|\underline{X}; \underline{\theta}))$

↓ ignore constant term

min. cost function $L(\underline{\theta}) = \frac{1}{N} \sum_{n=1}^N [-\ln(q(\underline{y}(n)|\underline{x}(n); \underline{\theta}))]$

↓ ch. 4.6: $q(\underline{y}|\underline{x}; \underline{\theta})$ for regression and classification?

l_2 -loss or l_1 -loss or categorical loss

5.4.1. 4.6.3 Semantic segmentation

pixelwise classification

categorical cross entropy loss:

$l(\underline{X}, \underline{Y}; \underline{\theta}) = \sum_{m=1}^M \sum_{n=1}^N [-y_{mn} \cdot \ln(\hat{y}_{mn}(\underline{X}, \underline{\theta}))]$ sum over all pixels

Problem: imbalanced classes

e.g (c=2) background and foreground with 90 % background and 10 % foreground pixels , 90 % loss function

for the foreground

→ $l(\cdot)$ cares more about the major class and less about the minor class but the minor class(foreground) is object of interest. → reduced segmentation accuracy for the minor class

Solutions:

- Weighted categorical CE loss
- region-based loss **Jaccard index only used for result evaluation not suitable as loss function for training**
- minimize loss, not max J or D for those indexes
- $|A| \in \mathbb{N}$, not differential with respect to $\underline{\theta}$
- \underline{y}_{mn} contains 0 or 1 as desired, but $\hat{\underline{y}}$ contains real numbers $\in (0, 1) \in \text{softmax}$
- adapted definition of J and D are necessary

Soft Jaccard and Dice loss

Let again $\mathbf{Y} = [\underline{y}_{mn}]_{mn}$ and $\hat{\mathbf{Y}} = [\hat{\underline{y}}_{mn}]_{mn}$. $\underline{y}_{mn} \in \{\underline{e}_1, \dots, \underline{e}_c\}$ is the one-hot coding and $\hat{\underline{y}}_{mn} \in \mathbb{R}^c$ is the c -class softmax output for the pixel (m, n) . The **soft Jaccard loss** to be minimized in training for image segmentation is

$$\begin{aligned}\underline{\alpha} &= \sum_{m=1}^M \sum_{n=1}^N \underline{y}_{mn} \odot \hat{\underline{y}}_{mn} = [\alpha_i] \in \mathbb{R}^c, \\ \underline{\beta} &= \sum_{m=1}^M \sum_{n=1}^N (\underline{y}_{mn} + \hat{\underline{y}}_{mn}) = [\beta_i] \in \mathbb{R}^c, \\ J_i &= \frac{\alpha_i + \epsilon}{\beta_i - \alpha_i + \epsilon}, \\ l(\mathbf{X}, \mathbf{Y}; \underline{\theta}) &= 1 - \frac{1}{c} \sum_{i=1}^c J_i\end{aligned}$$

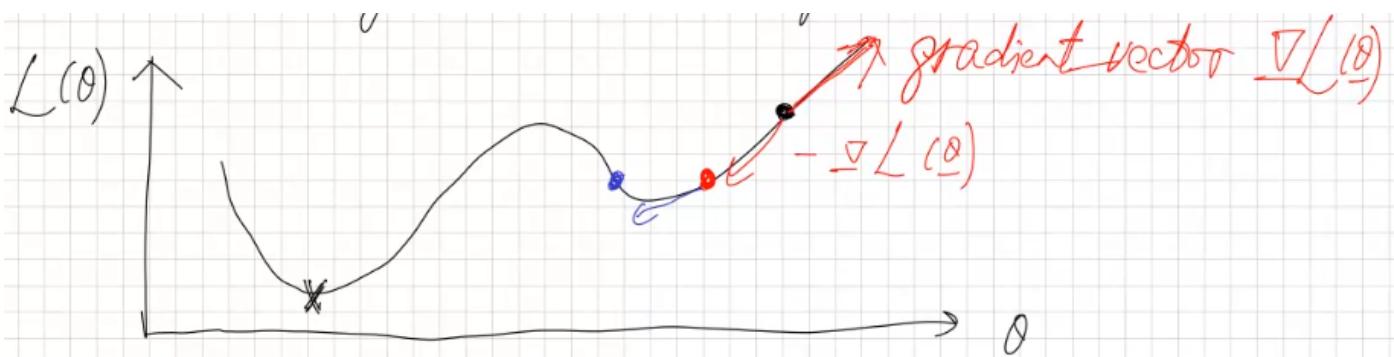
\odot is the elementwise multiplication. α_i and β_i represent arithmetic calculations of $|A_i \cap B_i|$ and $|A_i| + |B_i|$ for class i , respectively. J_i is the soft Jaccard index for class i where $\epsilon > 0$ is a suitable number to avoid 0/0 if $\alpha_i = \beta_i = 0$. $l(\mathbf{X}, \mathbf{Y}; \underline{\theta})$ is the soft Jaccard loss differentiable w.r.t. $\underline{\theta}$. The **soft Dice loss** is defined in a similar way.

For 3D segmentation, $\underline{\alpha}$ and $\underline{\beta}$ are calculated by three-dimensional sums over all pixels.

Very good for strongly imbalanced classes

5.5. Chapter 4.7 Training

- training set $train = \{\underline{x}, \underline{y}, 1 \leq n \leq N\}$
- cost function $L(\underline{\theta}) = \frac{1}{N} \sum_{n=1}^N l(\underline{x}(n), \underline{y}; \underline{\theta})$
- task: $\min L(\underline{\theta})$
- optimizer: optimization algorithm to solve the minimization task $L(\underline{\theta})$
No closed-form solution! Numerical minimization necessary, see AM (last part)
- in DL: gradient decent approach and variants(like hiking)**
need only 1. order derivative of $L(\underline{\theta})$



Update: $\underline{\theta}^{t+1} = \underline{\theta}^t - \gamma^t \underline{\nabla} L(\underline{\theta}^t)$

$t = 0, 1, \dots$: iteration index

$\gamma^t > 0$: step size

$\underline{\theta}^0$: initial guess

calculation of the gradient vector $\underline{\nabla} L(\underline{\theta})$ is non trivial

Chainrule of derivative (back propagation)

$$\frac{f(g(\theta))}{d\theta} = \frac{df}{dg} \cdot \frac{dg}{d\theta}$$

Layerindex L: $\frac{\partial L_{cost}(\underline{\theta})}{\partial w_{L,ij}} = \frac{\partial L_{cost}}{\partial \underline{x}_L} \cdot \frac{\partial \underline{x}_K}{\partial \underline{a}_L} \cdot \frac{\partial \underline{a}_L}{\partial w_{L,ij}} = \underbrace{\underline{J}_L(\underline{x}_L)}_{\underline{J}_L(\underline{a}_L)} \cdot \underbrace{\underline{J}_{\underline{x}_L}(\underline{a}_L)}_{\underline{J}_{\underline{a}_L}} \cdot \underbrace{\underline{J}_{\underline{a}_L}(w_{L,ij})}_{\underline{J}_{\underline{a}_{L-1}}(w_{L-1,ij})}$, Jacobi matrices see ch.

3.1

Notation: $\underline{J}_y(x) = \frac{\partial y}{\partial x}$

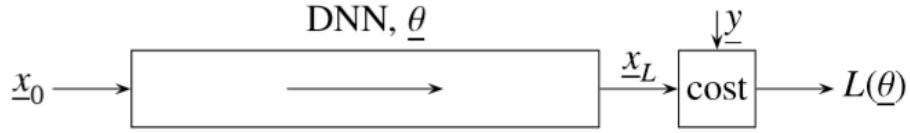
Layer $L-1$:

$$\frac{\partial L(\underline{\theta})}{\partial w_{L-1,ij}} = \frac{\partial L}{\partial \underline{a}_L} \cdot \frac{\partial \underline{a}_L}{\partial \underline{a}_{L-1}} \cdot \frac{\partial \underline{a}_{L-1}}{\partial w_{L-1,ij}} = \underbrace{\underline{J}_L(\underline{a}_L)}_{\underline{J}_L(\underline{a}_{L-1})} \cdot \underbrace{\underline{J}_{\underline{a}_L}(\underline{a}_{L-1})}_{\underline{J}_{\underline{a}_{L-1}}} \cdot \underbrace{\underline{J}_{\underline{a}_{L-1}}(w_{L-1,ij})}_{\underline{J}_{\underline{a}_{L-1}}(w_{L-1,ij})}$$

Layer 1: $\frac{\partial L(\underline{\theta})}{\partial w_{1,ij}} = \underline{J}_L(\underline{a}_L) \cdot \underline{J}_{\underline{a}_L}(\underline{a}_{L-1}) \cdot \dots \cdot \underline{J}_{\underline{a}_2}(\underline{a}_1) \cdot \underline{J}_{\underline{a}_1}(w_{1,ij})$

Forward pass vs. backward pass in DNN

Forward pass to calculate \underline{x}_L from \underline{x}_0 :



Backward pass or **backpropagation** of so called **error vectors** $\underline{\delta}_l^T := \mathbf{J}_L(\underline{a}_l) = \frac{\partial L(\underline{\theta})}{\partial \underline{a}_l} \in \mathbb{R}^{1 \times M_l}$:



For $l = L - 1, \dots, 1$

$$\underline{\delta}_l^T = \underline{\delta}_{l+1}^T \cdot \mathbf{J}_{\underline{a}_{l+1}}(\underline{x}_l) \mathbf{J}_{\underline{x}_l}(\underline{a}_l) = \boxed{\quad} \cdot \boxed{\quad}$$

$$\frac{\partial L(\underline{\theta})}{\partial w_{l,ij}} = \underline{\delta}_l^T \cdot \mathbf{J}_{\underline{a}_l}(w_{l,ij}) = \boxed{\quad} \cdot \boxed{\quad}$$

$$\frac{\partial L(\underline{\theta})}{\partial b_{l,i}} = \underline{\delta}_l^T \cdot \mathbf{J}_{\underline{a}_l}(b_{l,i}) = \boxed{\quad} \cdot \boxed{\quad}$$

Calculations for backpropagation on Slide 4-24 to 4-27

more about optimizer in ch.5

more about model in ch. 6-8

Batch gradient descent

calculate $\underline{\nabla} L(\underline{\theta}) = \frac{1}{N} \sum_{n=1}^N \underline{\nabla} l(\underline{x}(n), \underline{y}(n); \underline{\theta})$

for the whole training set D_{train}

Difficult for too large N!

e.g. MNIST: 60.000 training samples

ILSVRC: 1.2M training samples

→ not enough ram to keep D_{train} in memory

Solution: stochastic gradient descent (SGD), i.e. calculate $\underline{\nabla} L$ and update $\underline{\theta}$ for each minibatch, a block of B samples from d_{train} :

$$\underline{\theta}^{t+1} = \underline{\theta}^t - \underline{\nabla} L((t); \underline{\theta})|_{\underline{\theta}=\underline{\theta}_t}$$