Artificial Neural Networks: Lecture 3

Statistical classification by deep networks

Wulfram Gerstner
EPFL, Lausanne, Switzerland

Objectives for today:

- The cross-entropy error is the optimal loss function for classification tasks
- The sigmoidal (softmax) is the optimal output unit for classification tasks
- Multi-class problems and '1-hot coding'
- Under certain conditions we may interpret the output as a probability
- The rectified linear unit (RELU) for hidden layers

Reading for this lecture:

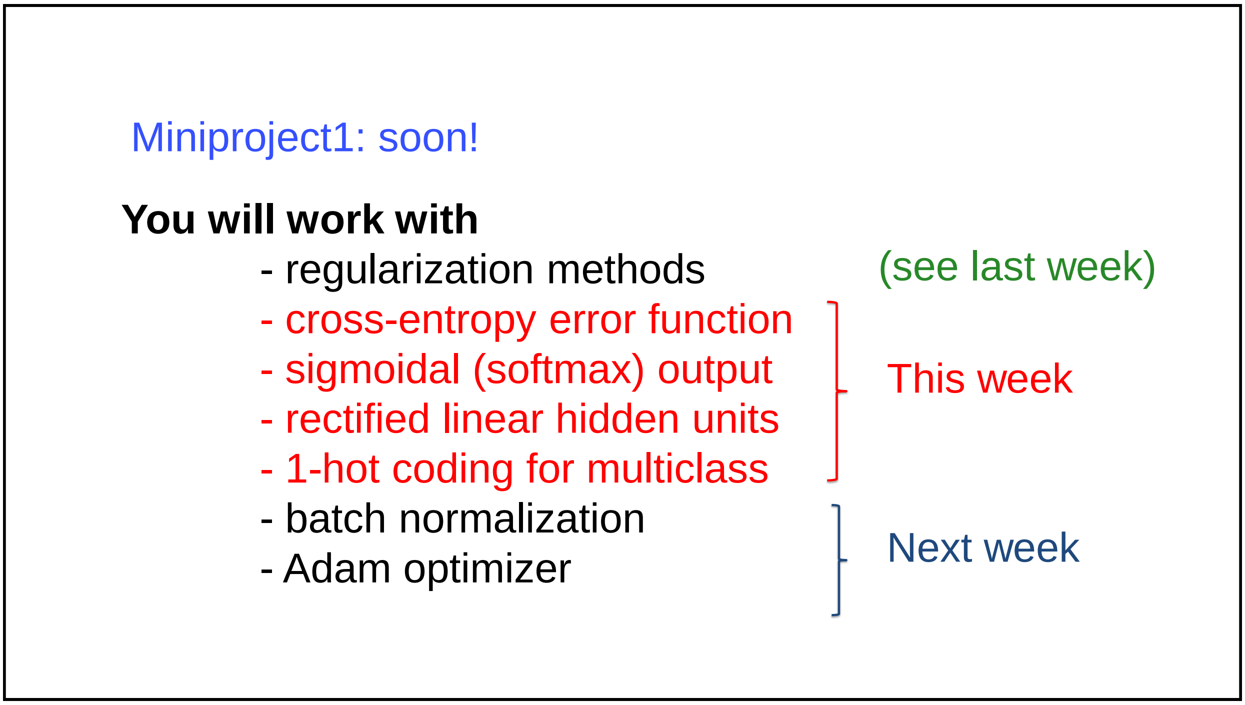
Bishop 2006, Ch. 4.2 and 4.3
Pattern recognition and Machine Learning

or

Bishop 1995, Ch. 6.7 – 6.9
Neural networks for pattern recognition

or

Goodfellow et al.,2016 Ch. 5.5, 6.2, and 3.13 of
Deep Learning



Miniproject1: soon!

You will work with

- regularization methods (see last week)
- cross-entropy error function
- sigmoidal (softmax) output
- rectified linear hidden units
- 1-hot coding for multiclass

This week

- batch normalization
- Adam optimizer

Next week

Previous slide.
Miniprojects must be done in groups of two students. Not one, not three.

The notions you need for the miniproject are introduced in parallel over the next two weeks, but you should start as soon as possible.

Review: Data base for Supervised learning (single output)

P data points $\{ \quad (\boldsymbol{x}^{\mu}, t^{\mu}) \quad , \qquad 1 \leq \mu \leq P \quad \};$

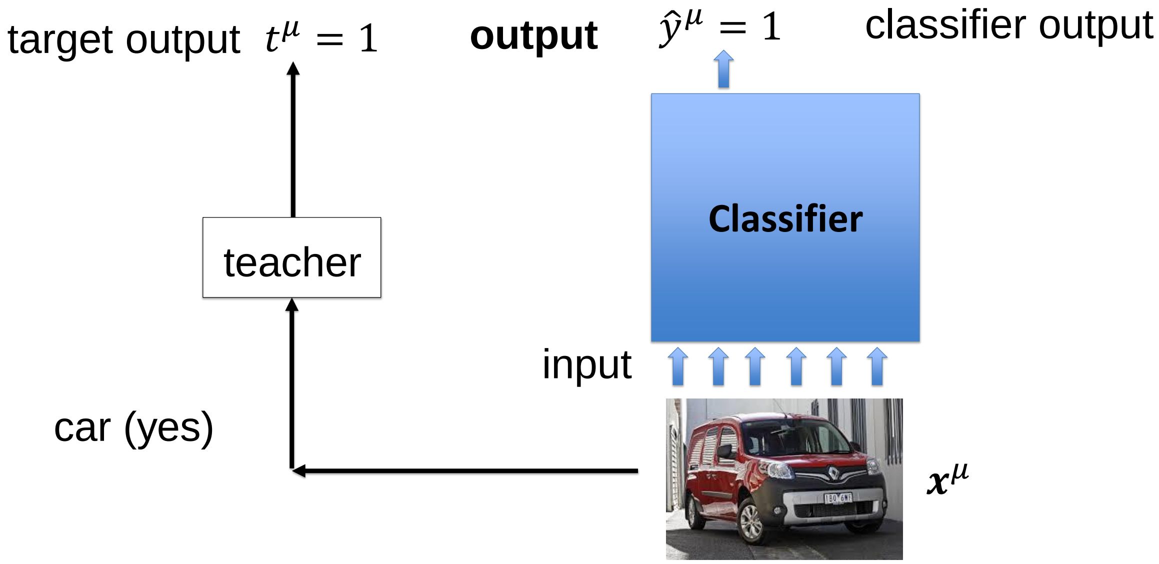
input target output

$t^{\mu} = 1$ car =yes

$t^{\mu} = 0$ car =no

Previous slide.
As we know from the previous lectures, we work in a supervised setting …

review: Supervised learning



target output $t^{\mu} = 1$

output

$\hat{y}^{\mu} = 1$

classifier output

teacher

Classifier

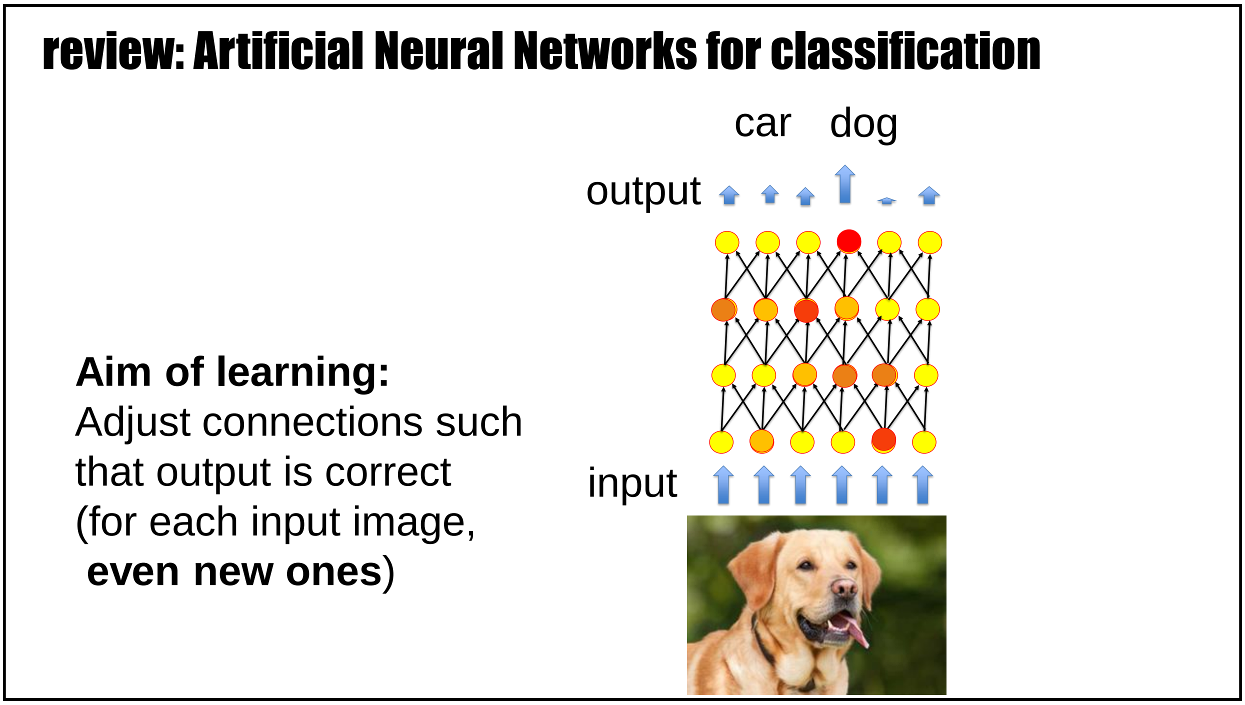
input

car (yes)

$\boldsymbol{x}^{\mu}$

Previous slide.

… and use the target information to adjust the parameters of our classifier …



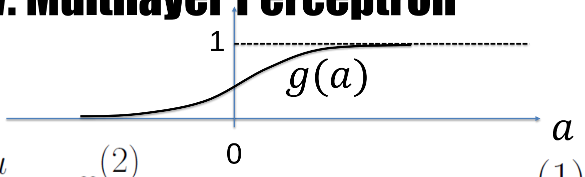
review: Artificial Neural Networks for classification

Aim of learning:
Adjust connections such that output is correct (for each input image, **even new ones**)

Previous slide.

… which is a multi-layer neural network in our case.

Review: Multilayer Perceptron



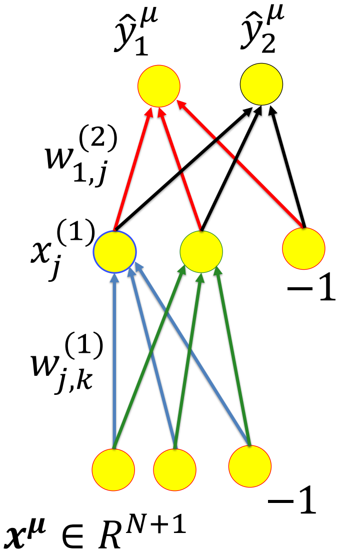
$$\hat{y}_i^{\mu} = x_i^{(2)} \quad (1)$$

$$= g^{(2)}[a_i^{(2)}] \quad (2)$$

$$= g^{(2)}[\sum_j w_{ij}^{(2)} x_j^{(1)}] \quad (3)$$

$$= g^{(2)}[\sum_j w_{ij}^{(2)} g^{(1)}(a_j^{(1)})] \quad (4)$$

$$= g^{(2)}[\sum_j w_{ij}^{(2)} g^{(1)}(\sum_k w_{jk}^{(1)} x_k^{\mu})] \quad (5)$$

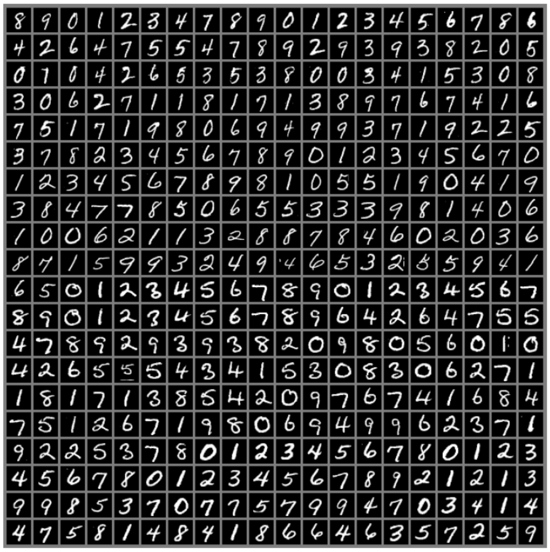


Previous slide.

The activity of output unit i for pattern number μ is denoted by $\hat{y}_1^{\mu}$.

Review: Example MNIST

MNIST data samples

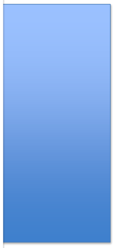


- images 28x28
- Labels: 0, …, 9
- 250 writers
- 60 000 images in training set

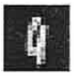
Picture: Goodfellow et al, 2016
Data base:
http://yann.lecun.com/exdb/mnist/

Previous slide.
As we have seen last week, all data is noisy

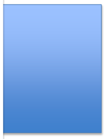
review: data base is noisy



9 or 4?



9 or 4?



- training data is always noisy
- the future data has different noise

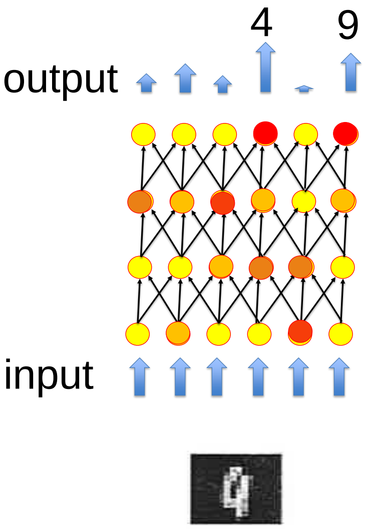
- Classifier must extract the essence
  → **do not fit the noise!!**

What might be a
9 for reader A
Might be a
4 for reader B

Previous slide.
Even the handwritten digits in the MNIST data base.

Question for today

May we interpret the outputs our network as a probability?



Previous slide.
In this lecture, we therefore reformulate the question of classification as follows:
Can we interpret an output activity $\hat{y}_4^\mu = 0.8$ as the probability of 80 percent that the pattern μ is a '4'?

Artificial Neural Networks: Lecture 3

Statistical Classification by Deep Networks

Wulfram Gerstner
EPFL, Lausanne, Switzerland

1. The statistical view: generative model

Previous slide.
A central notion for a probabilistic interpretation is the concept of a ‘generative model’.

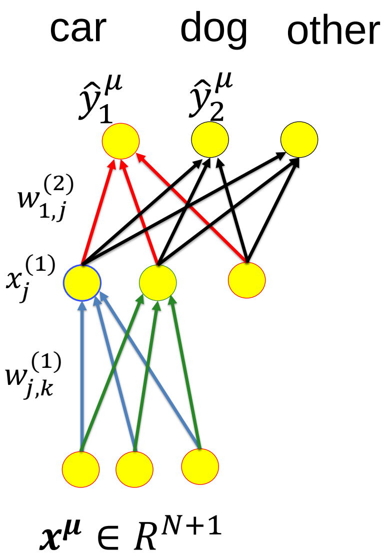
1. The statistical view

Idea:

interpret the output $\hat{y}_k^\mu$
as the **probability** that
the novel input pattern $\boldsymbol{x}^{\boldsymbol{\mu}}$
should be classified
as class k

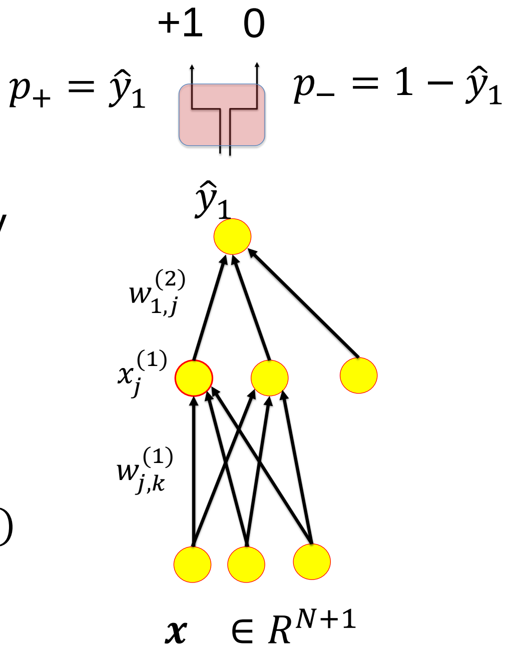
$\hat{y}_k^{\boldsymbol{\mu}} = \mathrm{P}(C_k|\boldsymbol{x}^{\boldsymbol{\mu}})$ pattern from data base

$\hat{y}_k = \mathrm{P}(C_k|\boldsymbol{x}\ \ )$ arbitrary novel pattern

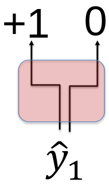


Previous slide.
The aim is to interpret the output of unit k
$\hat{y}_k = \mathrm{P}(C_k|\boldsymbol{x}\ \ )$
as the probability that the novel input pattern $\boldsymbol{x}$ belongs to class C_k
The notation is that of conditional probabilities
$\mathrm{P}(C_k|\boldsymbol{x}\ \ )$ is the probability of class C_k given $\boldsymbol{x}$.

1. The statistical view: single class



Take the output $\hat{y}_1$ and generate predicted labels $\hat{t}_1$ probabilistically



→ generative model for class label with $\hat{y}_1 = \mathrm{P}(C_1|\boldsymbol{x}) = \boldsymbol{P}(\hat{t}_1 = 1|\boldsymbol{x})$

↑ predicted label

Previous slide.

To enable such a probabilistic interpretation, we construct a generative model consisting of the neural network AND a probabilistic label generator.
The label generator outputs a

1 with probability $p_+ = \hat{y}_1$

And a

0 with probability $p_- = 1 - \hat{y}_1$

Then we can ask whether the label generated by the model is the (on average) the correct one. To see this we return to the data base of supervised learning.

Artificial Neural Networks: Lecture 3

Statistical Classification by Deep Networks

Wulfram Gerstner

EPFL, Lausanne, Switzerland

1. The statistical view: generative model
2. **The likelihood of data under a model**

Previous slide.
To compare the label generated by the model with the one in the data base of supervised learning, we take a maximum likelihood approach.

2. The likelihood of a model (given data)

Overall aim:
What is the probability that my set of P data points

$$\{ \quad (\boldsymbol{x}^{\mu}, t^{\mu}) \quad , \qquad 1 \le \mu \le P \quad \};$$

could have been generated by my model?

Previous slide.
Specifically we ask:
What is the probability that all my data points (input patterns and labels) could have been generated by my model?

2. The likelihood of a model

Detour:

forget about labeled data, and just think of input patterns

What is the probability that a set of P data points

$$\{\boldsymbol{x}^k ; 1 \leq \mathrm{k} \leq P \};$$

could have been generated by my model?

Previous slide.
Since it is a bit difficult to think about labels as a statistical process, let us consider first the classical problem of generating (unlabeled) data points.

2. Example: Gaussian distribution

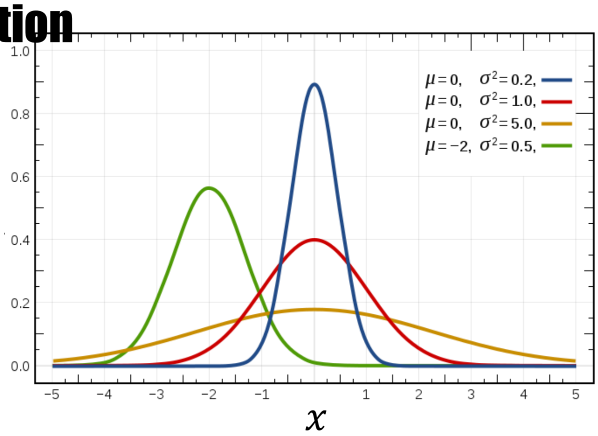
$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} exp\left\{\frac{-(x-\mu)^2}{2\sigma^2}\right\}_1$$

this depends on 2 parameters

$$\{w_{1,} w_{2,}\} = \{\mu, \sigma\}$$

center

width



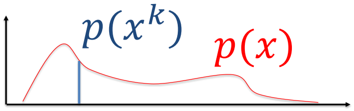
https://en.wikipedia.org/wiki/Gaussian_function#/media/

Previous slide.
Suppose that we know that the data comes from a Gaussian distribution. However, we do not know the mean (center) and the standard deviation (width) of the distribution.

2. Random Data Generation Process

Probability that a random data generation process draws one sample k with value x^k is

$$\sim p(x^k)$$



Example: for the specific case of the Gaussian

$$p(x^k) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{\frac{-(x^k-\mu)^2}{2\sigma^2}\right\}$$

What is the probability to generate P data points?

Previous slide.
In general, the distribution is not a Gaussian, but some function $p(x)$.
We observe P data points and ask:
What is the probability that the specific set of P data points COULD HAVE BEEN GENERATED by a random data generation process which draws data from p(x)?
The probability that you would draw a data point x in the range

$$x^k - \frac{\Delta x}{2} < x^k \leq x^k + \frac{\Delta x}{2}$$

is

$$P = p(x^k)\Delta x$$

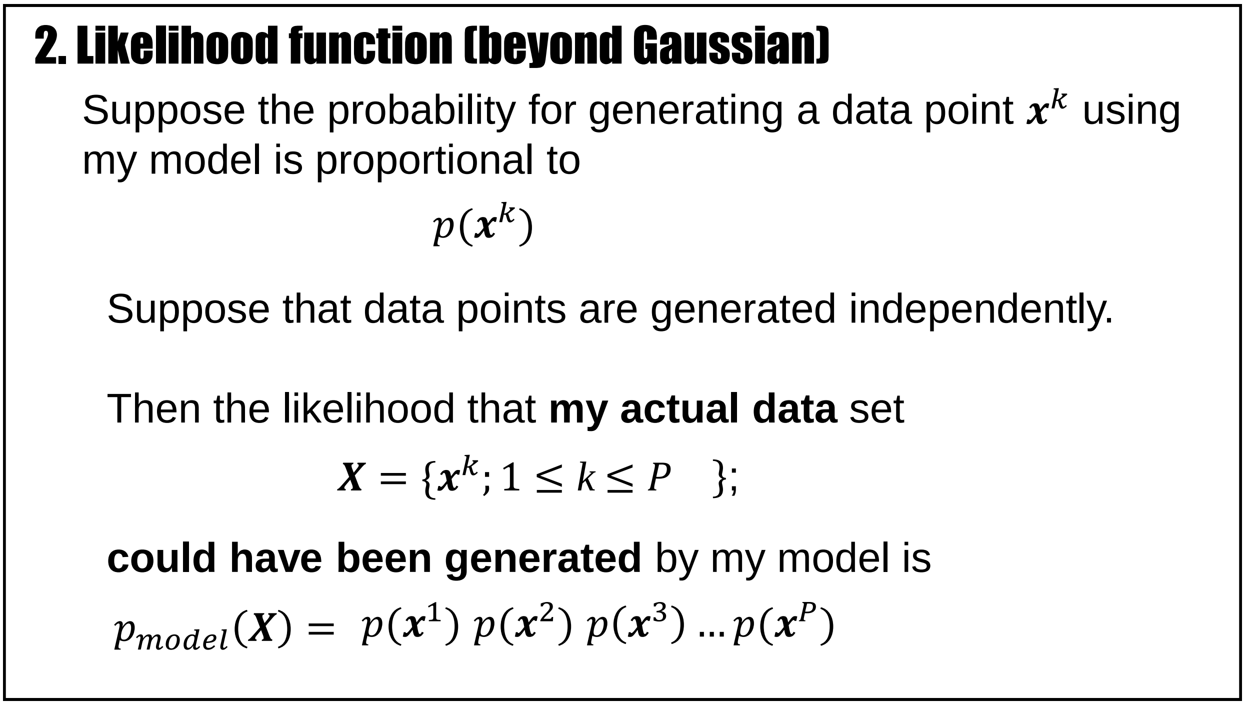
which is correct in the limit that $\Delta x \to 0$.

Therefore the probability is to draw the point x^k is PROPORTIONAL to $\sim p(x^k)$

The aim is now to generate not just a single one, but all P data points.

Blackboard 1:
generate P data points

Your notes.



2. Likelihood function (beyond Gaussian)

Suppose the probability for generating a data point $\boldsymbol{x}^k$ using my model is proportional to

$$p(\boldsymbol{x}^k)$$

Suppose that data points are generated independently.

Then the likelihood that **my actual data** set

$$\boldsymbol{X} = \{\boldsymbol{x}^k; 1 \leq k \leq P \quad \};$$

could have been generated by my model is

$$p_{model}(\boldsymbol{X}) = \; p(\boldsymbol{x}^1)\; p(\boldsymbol{x}^2)\; p(\boldsymbol{x}^3) \ldots p(\boldsymbol{x}^P)$$

Previous slide.
Under the assumption that data points are generated independently, the likelihood that total data set of P data points could have been generated by my model is

$$p(\boldsymbol{x}^1)\; p(\boldsymbol{x}^2)\; p(\boldsymbol{x}^3) \ldots p(\boldsymbol{x}^P)$$

Here I use the term *likelihood* in a broad sense defined as a quantity that is proportional to the probability. The narrow definition comes on the next slide.

2. Maximum Likelihood (beyond Gaussian)

$$p_{model}(\boldsymbol{X}) = \quad p(\boldsymbol{x}^1)\, p(\boldsymbol{x}^2)\, p(\boldsymbol{x}^3) \ldots p(\boldsymbol{x}^P)$$

BUT this likelihood depends on the parameters of my model

$$p_{model}(\boldsymbol{X}) = p_{model}\left(\boldsymbol{X} \middle| \left\{w_{1,} w_{2,} \ldots w_{n,}\right\}\right)$$

parameters

Choose the parameters such that the likelihood is maximal!

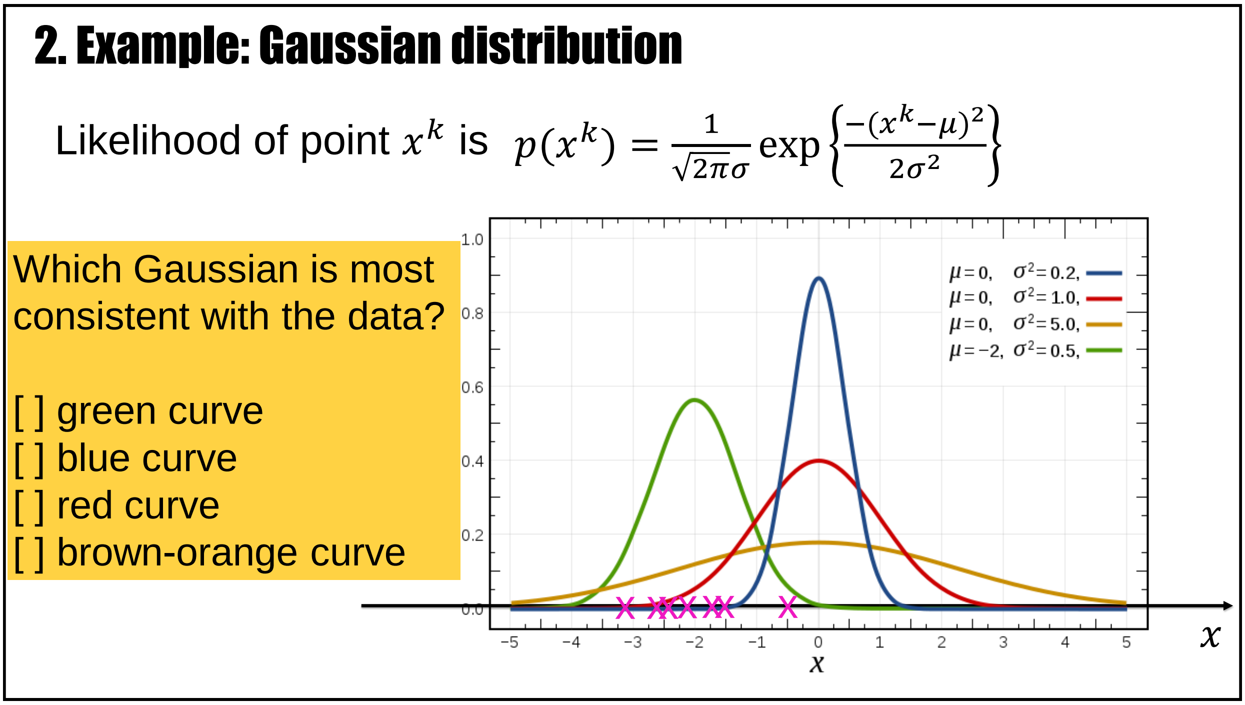
Previous slide.

The expression $p_{model}\left(\boldsymbol{X} \middle| \left\{w_{1,} w_{2,} \ldots w_{n,}\right\}\right)$

as a function of the model parameters is called the *likelihood function* (in the narrow sense).

The aim is now to choose the parameters of the model such that the likelihood that the data COULD HAVE BEEN GENERATED by the model is maximal.

This optimization procedure is called the Maximum-Likelihood (ML) approach.



2. Example: Gaussian distribution

Likelihood of point x^k is $p(x^k) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{\frac{-(x^k - \mu)^2}{2\sigma^2}\right\}$

Which Gaussian is most consistent with the data?

[ ] green curve
[ ] blue curve
[ ] red curve
[ ] brown-orange curve

Previous slide.
For example, it is very unlikely that the data points (pink crosses) could have been generated by the blue distribution.

2. Example: Gaussian

$$p_{model}(\boldsymbol{X}) = \quad p(\boldsymbol{x}^1)\, p(\boldsymbol{x}^2)\, p(\boldsymbol{x}^3) \ldots p(\boldsymbol{x}^P)$$

The likelihood depends on the 2 parameters of my Gaussian

$$p_{model}(\boldsymbol{X}) = p_{model}\left(\boldsymbol{X}|\{w_{1,}w_2\}\right)$$

$$p_{model}(\boldsymbol{X}) = p_{model}(\boldsymbol{X}|\{\mu, \sigma\})$$

Exercise 1 NOW! (8 minutes): you have P data points
Calculate the **optimal choice** of parameter μ:
To do so maximize $p_{model}(\boldsymbol{X})$ with respect to μ

Your notes

Blackboard 2:
Gaussian: best parameter choice for center

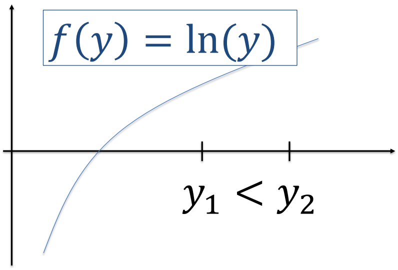
Your notes.

2. Maximum Likelihood (general)

Choose the parameters such that the likelihood

$$p_{model}\left(\boldsymbol{X}|\{w_{1,}w_{2,} \dots w_{n,}\}\right) = p(\boldsymbol{x}^1)\ p(\boldsymbol{x}^2)\ p(\boldsymbol{x}^3) \dots p(\boldsymbol{x}^P)$$

is maximal



Note:
Instead of maximizing

$$p_{model}(\boldsymbol{X}|param)$$

you can also maximize

$$\ln(p_{model}(\boldsymbol{X}|param))$$

Previous slide.

The idea is to use, amongst all the possible models, the model with the highest likelihood that it could have generated the actual observed data. The family of models is characterized by parameters.
In the case of the Gaussian, the parameters are the center and the width of the Gaussian.
In our case, the parameters will be the weight of the neural network.

Whatever you choose as a family of models, you want to maximize the likelihood

$$p_{model}\left(\boldsymbol{X}|\{w_{1,}w_{2,} \dots w_{n,}\}\right)$$

that the observed data could have been generated by your model.

Because the logarithm is a monotone function, the parameter values you find by maximizing this likelihood are the same as the parameter values you would find if you maximize

$$\ln[p_{model}\left(\boldsymbol{X}|\{w_{1,}w_{2,} \dots w_{n,}\}\right)]$$

2. Maximum Likelihood (general)

Choosing the parameters such that the likelihood

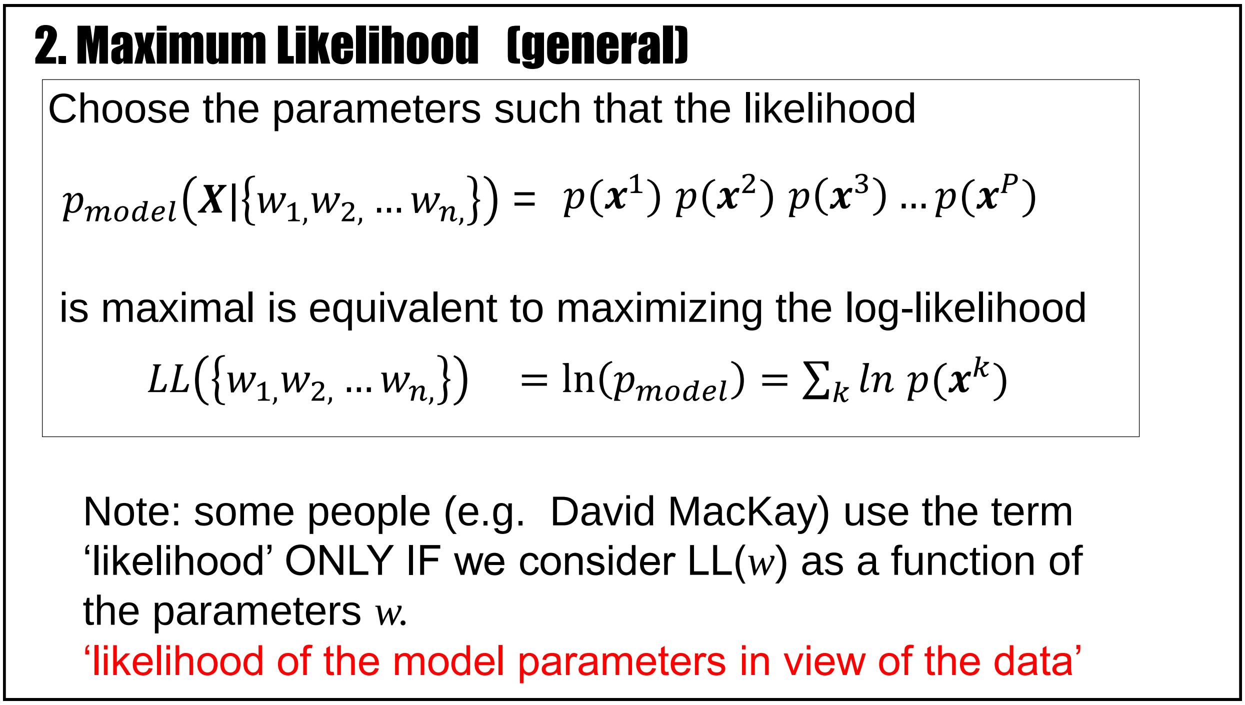
$$p_{model}\left(\boldsymbol{X}|\{w_{1,}w_{2,}\dots w_{n,}\}\right) = p(\boldsymbol{x}^1)\,p(\boldsymbol{x}^2)\,p(\boldsymbol{x}^3)\dots p(\boldsymbol{x}^P)$$

is maximal is equivalent to maximizing the log-likelihood

$$LL\left(\{w_{1,}w_{2,}\dots w_{n,}\}\right) \quad = \ln(p_{model}) = \sum_k ln\; p(\boldsymbol{x}^k)$$

“Maximize the likelihood that the given data could have been generated by your model”
(even though you know that the data points were generated by a process in the real world that might be very different)

Previous slide.
Note that we make no claim that the data actually was generated by your model. The data may not be Gaussian, and you still try to fit it by a Gaussian.



2. Maximum Likelihood (general)

Choose the parameters such that the likelihood

$$p_{model}(\boldsymbol{X}|\{w_{1,}w_{2,}\dots w_{n,}\}) = p(\boldsymbol{x}^1)\, p(\boldsymbol{x}^2)\, p(\boldsymbol{x}^3) \dots p(\boldsymbol{x}^P)$$

is maximal is equivalent to maximizing the log-likelihood

$$LL(\{w_{1,}w_{2,}\dots w_{n,}\}) = \ln(p_{model}) = \sum_k ln\, p(\boldsymbol{x}^k)$$

Note: some people (e.g. David MacKay) use the term 'likelihood' ONLY IF we consider LL(w) as a function of the parameters w.

'likelihood of the model parameters in view of the data'

Previous slide.

The likelihood is a function of the model parameters.

Finding the maximum of the likelihood gives you the best model parameters in the sense of 'maximum likelihood'.

Note: The likelihood as a function of model parameters is not normalized to one and therefore you should not think of it as a probability or a probability density.

Artificial Neural Networks: Lecture 3
Statistical Classification by Deep Networks

Wulfram Gerstner
EPFL, Lausanne, Switzerland

1. The statistical view: generative model
2. The likelihood of data under a model
3. **Application to artificial neural networks**

Previous slide.

We now apply to concept of maximum likelihood to artificial neural networks

3. The likelihood of data under a neural network model

Overall aim:
What is the likelihood that my set of P data points

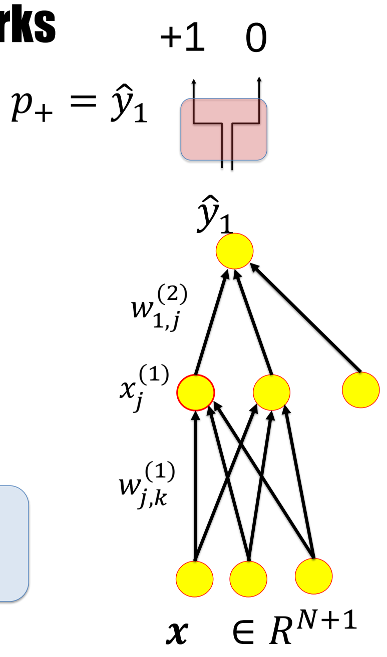
$$\{ \quad (\boldsymbol{x}^{\mu}, t^{\mu}) \quad , \qquad 1 \leq \mu \leq P \quad \};$$

could have been generated by my model?

Previous slide.

We now ask: what is the likelihood that the P pairs (input, target) that we have in the training base **could have been** generated by my neural network?

3. Maximum Likelihood for neural networks



Blackboard 3:

Likelihood of P input-output pairs

Previous slide.

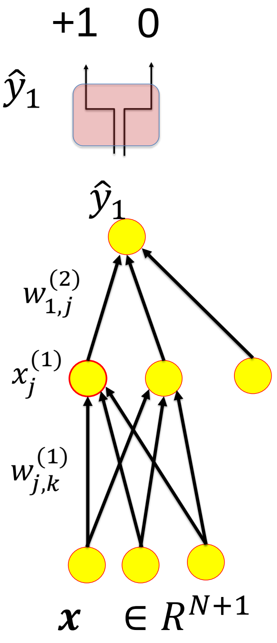
To analyze this we return to the generative model where the neural network output is interpreted as the probability to generate a label 1 or 0.

Blackboard 3:
Likelihood of P input-output pairs

Your notes.

3. Maximum Likelihood for neural networks

$p_+ = \hat{y}_1$



Minimize the negative log-likelihood

$$E(w) = -LL = -\ln(p_{model})$$

parameters= all weights, all layers

$$E(w) = -\sum_\mu [t^\mu \, ln \, \hat{y}^\mu + (1-t^\mu)\ln(1-\hat{y}^\mu)]$$

Previous slide.
The advantage of working with the negative log-likelihood, -LL, rather than the likelihood itself, is that the resulting error function (loss function) is just a sum over all data points.

It is called the 'cross-entropy' error function because of its similarity with quantities such as the entropy of a binary distribution or the Kullback-Leibler divergence that frequently occur in information theory; see Section 5 below.

The arguments so far give us the following result:
If we want to interpret the output as a probability, the best parameters (in the sense of maximum likelihood) are those that minimizing the cross-entropy error function.

In other words: the quadratic error function is not the best one for classification tasks if we aim for a statistical interpretation. For a statistical interpretation, we need to use the cross-entropy error function.

3. Cross-entropy error function for neural networks

Suppose we minimize the cross-entropy error function

$$E(w) = -\sum_{\mu}[t^{\mu}\, ln\, \hat{y}^{\mu} + (1 - t^{\mu})\ln(1 - \hat{y}^{\mu})]$$

Can we be sure that the output $\hat{y}^{\mu}$ will represent the probability?

Intuitive answer: **No, because**

A We will need enough data for training

(not just 10 data points for a complex task)

B We need a sufficiently flexible network

(not a simple perceptron for XOR task)

Previous slide.

Now we ask the inverse question:
Let us start with the cross-entropy error function. If we have found the parameters that minimize the cross-entropy error function, does that guarantee that the output is the probability?

The answer has to be negative, because:

A if we do not have enough data points, the best error function cannot help us.

B if the network is just a simple perceptron, it is most likely not be flexible enough to solve the task at hand.

In the following, we will therefore assume that
A we have always '**enough data'** and B the network is **'flexible enough'**.
The mathematical arguments on the blackboard should show HOW these assumptions are used to derive some strong conclusions.

3. Output = probability ?

Suppose we minimize the cross-entropy error function

$$E(w) = -\sum_{\mu}[t^{\mu}\, ln\, \hat{y}^{\mu} + (1-t^{\mu})\ln(1-\hat{y}^{\mu})]$$

Assume

A We have enough data for training

B We have a sufficiently flexible network

Blackboard 4:

From Cross-entropy to output probabilities

Your notes.

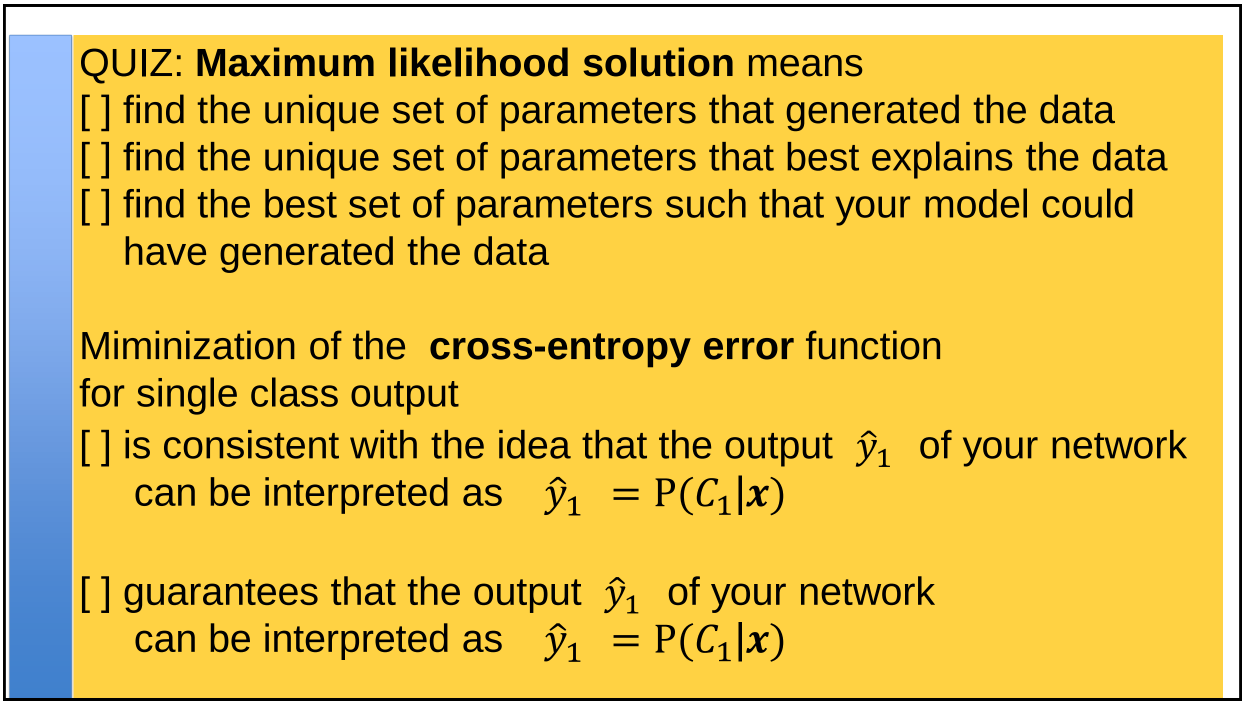
Blackboard 4: From Cross-entropy to output probabilities

Previous slide/Blackboard calculations.

The calculations with discrete bins Δx show that the notions ‘enough examples’ and ‘flexible enough’ are linked to each other: we need enough data samples in each bin to reliably estimate the fraction of positive examples in a bin; and the network must have enough flexibility to output for each bin a different value.

In neural network applications we do not have discrete inputs, but the logic is the same: the flexibility of the network per unit distance (controlled by regularization) must match the number of data points per unit distance, and analogously in high dimensions.

Minimization of the crossentropy guarantees that the output is a probability only in the limit of an infinite number of data points in a network of arbitrary flexibility.



QUIZ: **Maximum likelihood solution** means

[ ] find the unique set of parameters that generated the data

[ ] find the unique set of parameters that best explains the data

[ ] find the best set of parameters such that your model could have generated the data

Miminization of the **cross-entropy error** function for single class output

[ ] is consistent with the idea that the output $\hat{y}_1$ of your network can be interpreted as $\hat{y}_1 = \mathrm{P}(C_1|\boldsymbol{x})$

[ ] guarantees that the output $\hat{y}_1$ of your network can be interpreted as $\hat{y}_1 = \mathrm{P}(C_1|\boldsymbol{x})$

Your notes.

Artificial Neural Networks: Lecture 3

Statistical Classification by Deep Networks

Wulfram Gerstner
EPFL, Lausanne, Switzerland

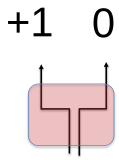
1. The statistical view: generative model
2. The likelihood of data under a model
3. Application to artificial neural networks
4. **Sigmoidal as a natural output function**

Previous slide.
In this part we show that the sigmoidal function enables a nice probabilistic interpretation of the neuronal activities.

4. Why sigmoidal output ? – single class

$\hat{y}_1 = \mathrm{P}(C_1|\boldsymbol{x}) = \boldsymbol{P}(\hat{t}_1 = 1|\boldsymbol{x})$

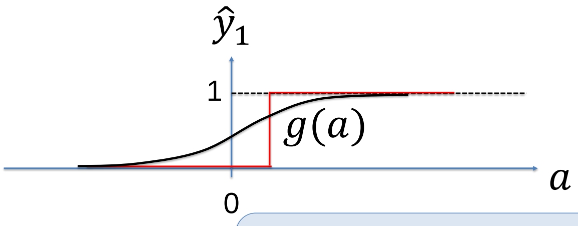
$p_+ = \hat{y}_1$



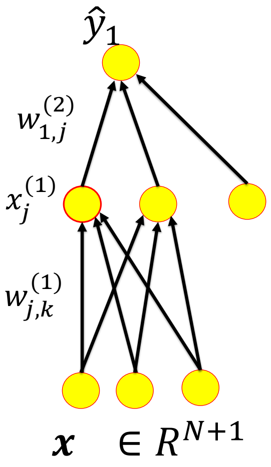
$p_- = 1 - \hat{y}_1$

Observations (single-class):

- Probability must be between 0 and 1
- Inutitively: smooth is better

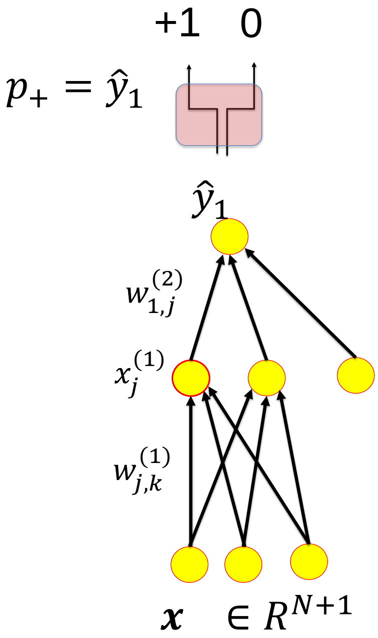


Blackboard 5:
derive optimal sigmoidal

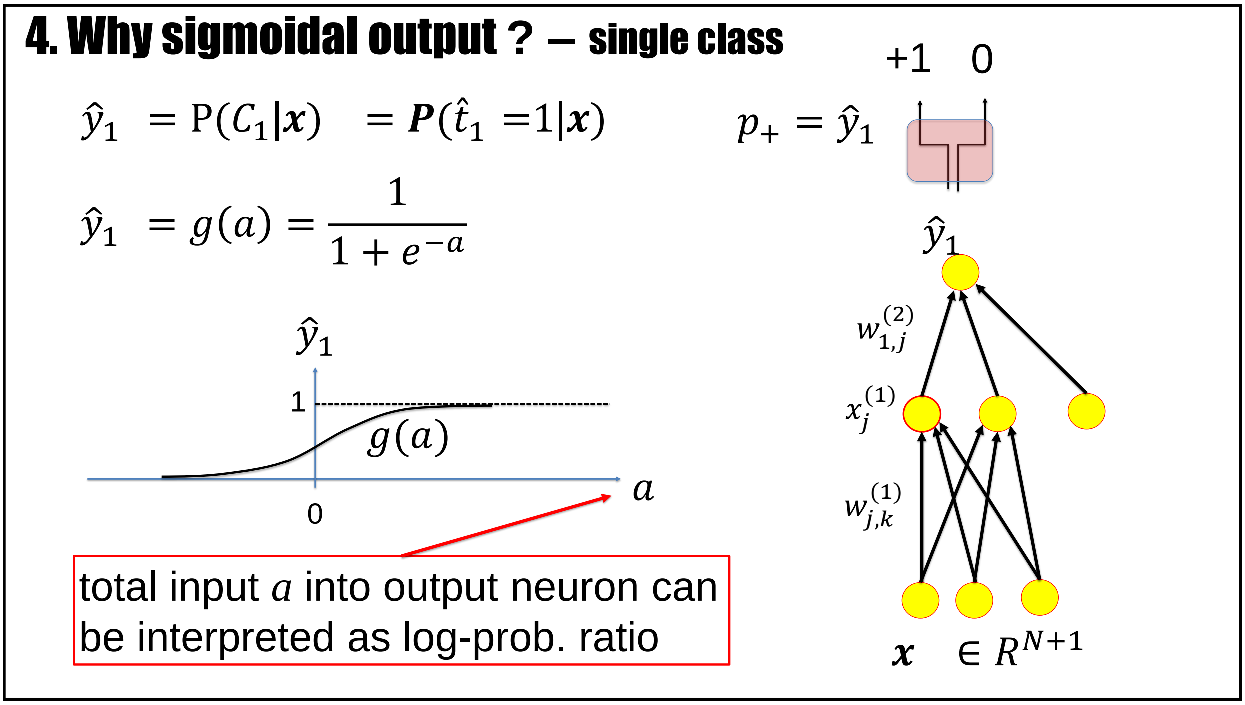


Previous slide.

Blackboard 5:
derive optimal sigmoidal



Your notes.



4. Why sigmoidal output ? – single class

$$\hat{y}_1 = \mathrm{P}(C_1|\boldsymbol{x}) = \boldsymbol{P}(\hat{t}_1 = 1|\boldsymbol{x})$$

$$\hat{y}_1 = g(a) = \frac{1}{1+e^{-a}}$$

$$p_+ = \hat{y}_1$$

total input a into output neuron can be interpreted as log-prob. ratio

Previous slide.

The total input (activation variable *a*) of the output neuron can be interpreted as the logarithm of the fraction p(x,C)/p(x,\C)

```
      probability p(x,C)      that the input is x and belongs to the class
 ln -----------------------------------------------------------------------------------------
      probability p(x, \C) that the input is x and does not belong to the class
```

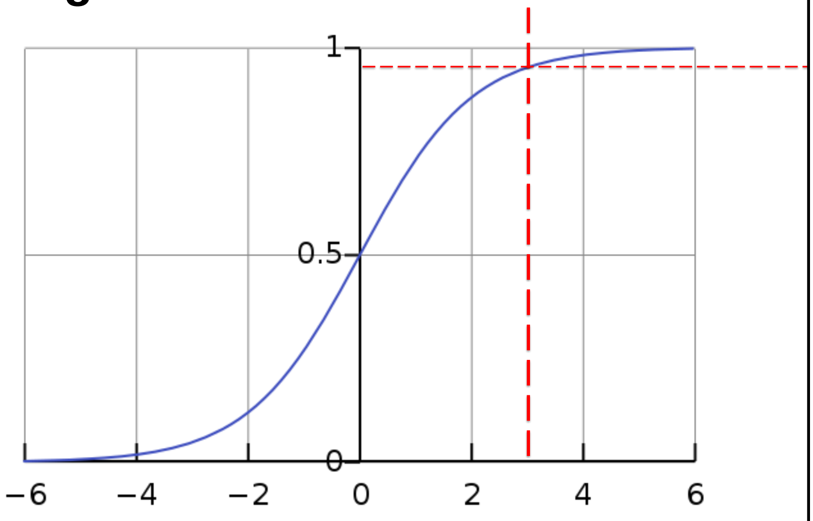
called the log-probability ratio. Here \C means: does NOT belong to class C.
People interpret the log-probabilities as a difference between
 ‘evidence in favor for assignment to class ’
 ‘evidence against assignment to class ’ .

Exercise this week: compare with ln[p(C|x)/p(\C|x)]

4. sigmoidal output = logistic function

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

Rule of thumb:
for *a= 3: g(3) =0.95*
for *a=-3: g(-3)=0.05*



https://en.wikipedia.org/wiki/Logistic_function

Previous slide.

Above an activation value of $a = 3$ the probability to generate a 1 in the output is above 95 percent.

Thus the most likely output is ‘yes, this input belongs to the class’.

For an activity of zero, the probability is exactly 50 percent:
There is as much evidence for and against the assignment of this input to the class.

Artificial Neural Networks: Lecture 3
Statistical Classification by Deep Networks

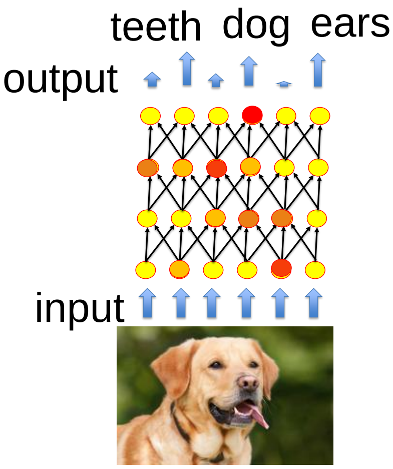
Wulfram Gerstner
EPFL, Lausanne, Switzerland

1. The statistical view: generative model
2. The likelihood of data under a model
3. Application to artificial neural networks
4. Sigmoidal as a natural output function
5. **Multi-class problems**

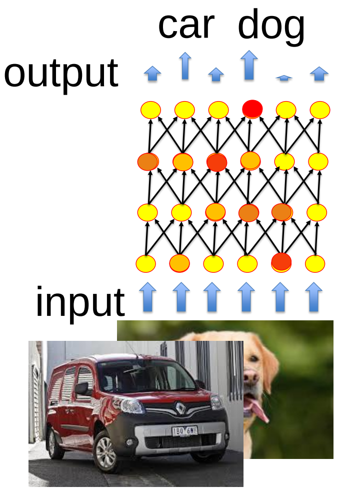
Previous slide.
So far we worked with probabilities for a single class C, such as ‘car’ versus ‘not car’, represented by a neural network with a SINGLE output.

5. Multiple Classes

multiple attributes



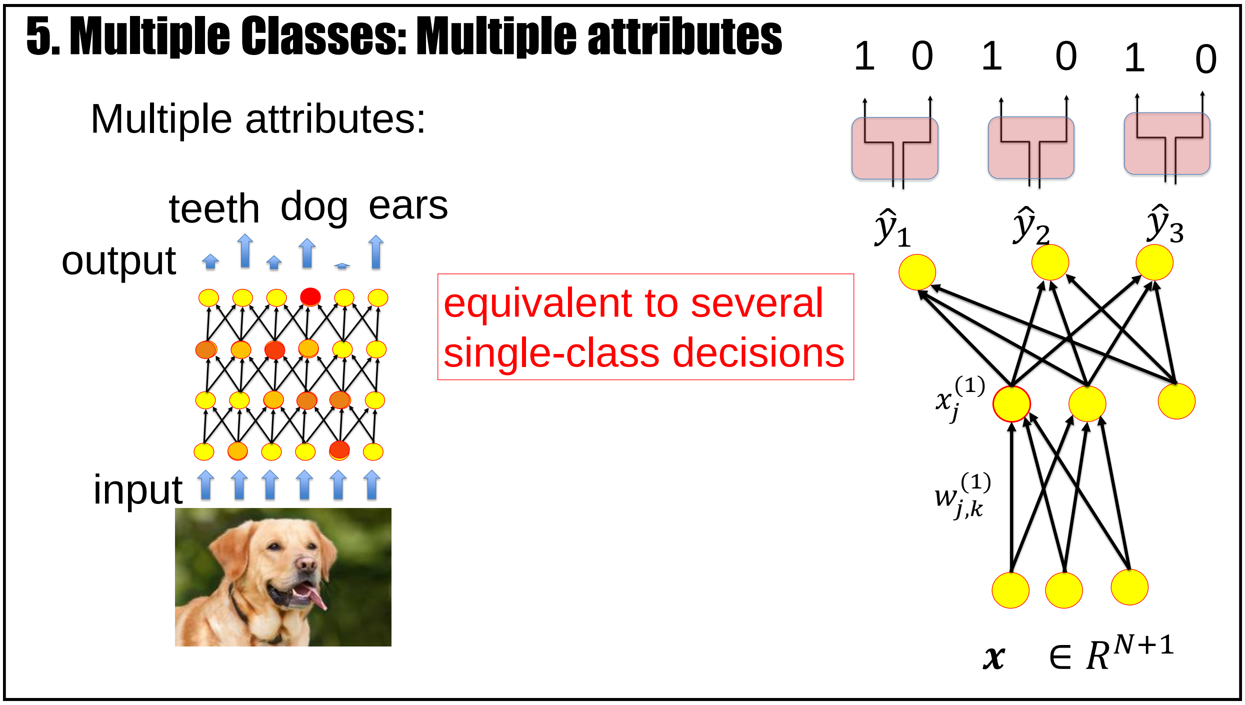
mutually exclusive classes



Previous slide.
In the following we consider a network with multiple outputs. We need to distinguish between two different paradigms.

A (left): the outputs refer to attributes. For example a dog picture can show teeth, and ears, and the dog. Therefore several outputs can be active at the same time.

B (right): the outputs refer to mutually exclusive classes. For example, an image can be classified as either a car or a dog or a house, but not two at the same time. [This is an important assumption which implies that I we exclude the case that the image could show one car and two dogs.]



5. Multiple Classes: Multiple attributes

Multiple attributes:

equivalent to several single-class decisions

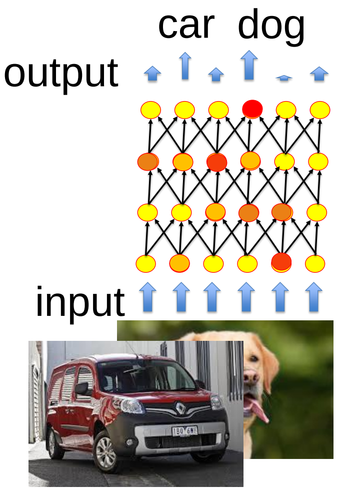
Previous slide.

The case of multiple attributes can be treated as several single-class decisions that are taken in parallel. The image can contain ears or not ears; teeth or not teeth, … . The fact that we see an ear does not exclude the fact that we see on the same image also some teeth.

5. Multiple Classes: Mutuall exclusive classes

mutually exclusive classes

either car or dog:
only one can be true
→
outputs interact



Previous slide.

For mutually exclusive classes, the situation is different. We assume that the image contains one class at a time: either a car, or a dog, or a tree, or a house, but not several items at the same time.

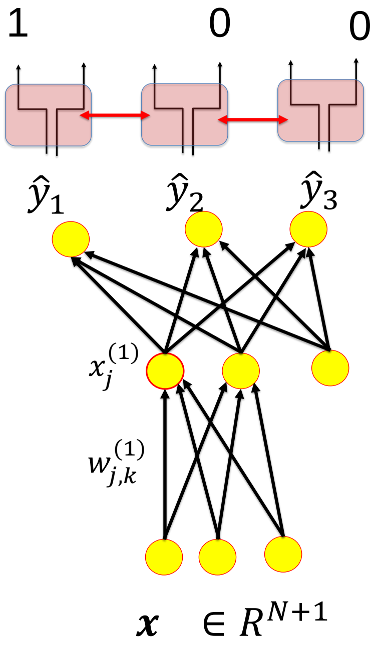
This implies that the outputs must interact. If one of the output shows 'true' the other must show 'not true'.

5. Exclusive Multiple Classes

$\hat{y}_1 \;\; = \mathrm{P}(C_1|\boldsymbol{x}) \;\;\; = \boldsymbol{P}(\hat{t}_1 \; =1|\boldsymbol{x})$

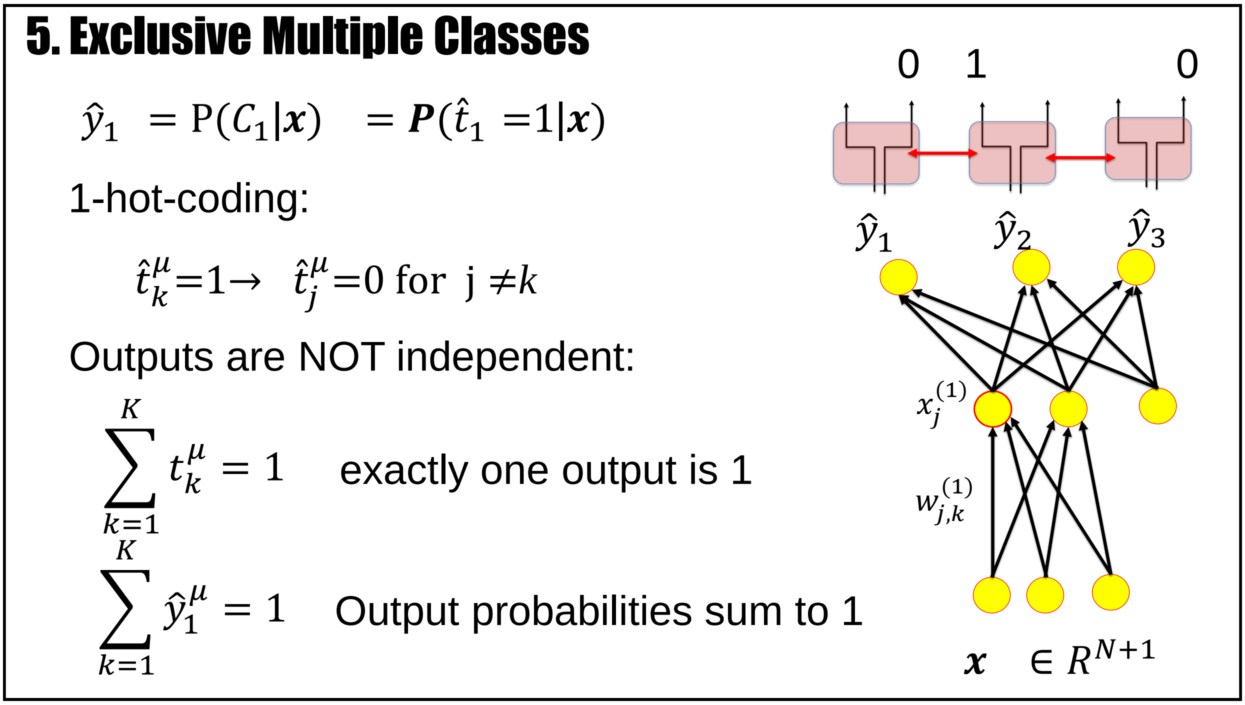
1-hot-coding:

$\hat{t}_k^{\mu}=1 \rightarrow \;\; \hat{t}_j^{\mu}=0 \text{ for } \; \mathrm{j} \neq k$



Previous slide.

In the case of mutually exclusive classes, the database of supervised learning has labels that reflect ‘one-hot coding’: For each input $\boldsymbol{x}$ exactly one of the target values is 1 and all the others are zero.



5. Exclusive Multiple Classes

$$\hat{y}_1 \;= \mathrm{P}(C_1|\boldsymbol{x}) \quad = \boldsymbol{P}(\hat{t}_1 \;=1|\boldsymbol{x})$$

1-hot-coding:

$$\hat{t}_k^{\mu}=1 \rightarrow \quad \hat{t}_j^{\mu}=0 \text{ for } \mathrm{j} \neq k$$

Outputs are NOT independent:

$$\sum_{k=1}^{K} t_k^{\mu} = 1$$ exactly one output is 1

$$\sum_{k=1}^{K} \hat{y}_1^{\mu} = 1$$ Output probabilities sum to 1

Previous slide.

Similarly, an interpretation of the networks outputs as probabilities implies that the outputs of the network must sum to one.

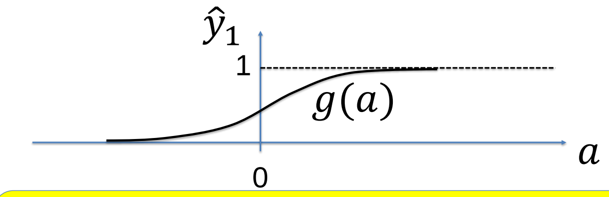
$$\sum_{k=1}^{K} \hat{y}_1^{\mu} = 1$$

5. Why sigmoidal output ?

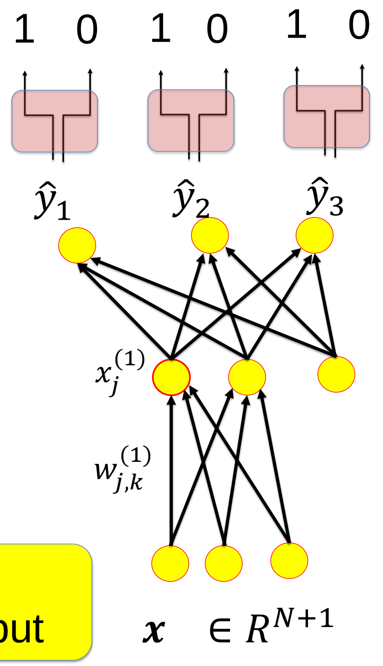
$$\hat{y}_1 = \mathrm{P}(C_1|\boldsymbol{x}) = \boldsymbol{P}(\hat{t}_1 = 1|\boldsymbol{x})$$

Observations (multiple-classes):

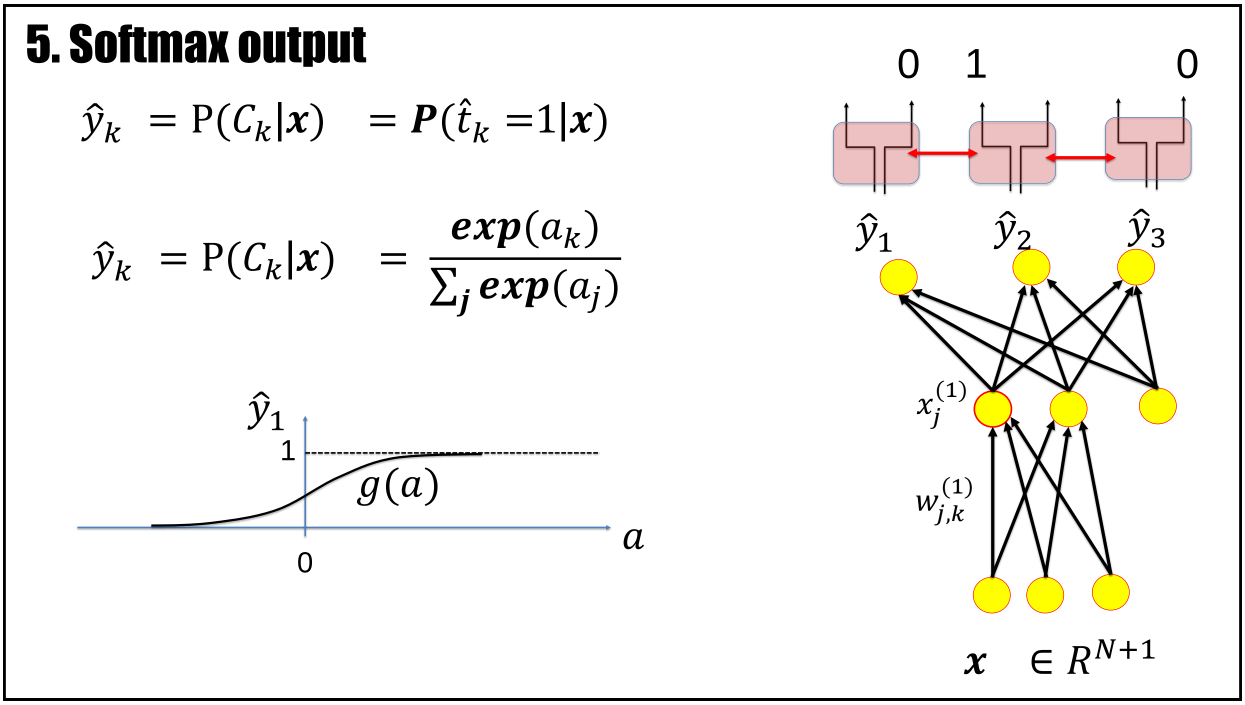
- Probabilities must sum to one!



Exercise this week!
derive softmax as optimal multi-class output



Your notes.
In the exercises this week, you will show that the conditions that
(i) outputs are probabilities
(ii) probabilities add up to one,
imply the 'softmax' output function.



5. Softmax output

$$\hat{y}_k = \mathrm{P}(C_k|\boldsymbol{x}) = \boldsymbol{P}(\hat{t}_k = 1|\boldsymbol{x})$$

$$\hat{y}_k = \mathrm{P}(C_k|\boldsymbol{x}) = \frac{\boldsymbol{exp}(a_k)}{\sum_j \boldsymbol{exp}(a_j)}$$

Previous slide.

A generative probabilistic model for mutually exclusive classes (‘1-hot-coding’) implies a neural network where the output neurons interact in the form of a ‘softmax’ function.

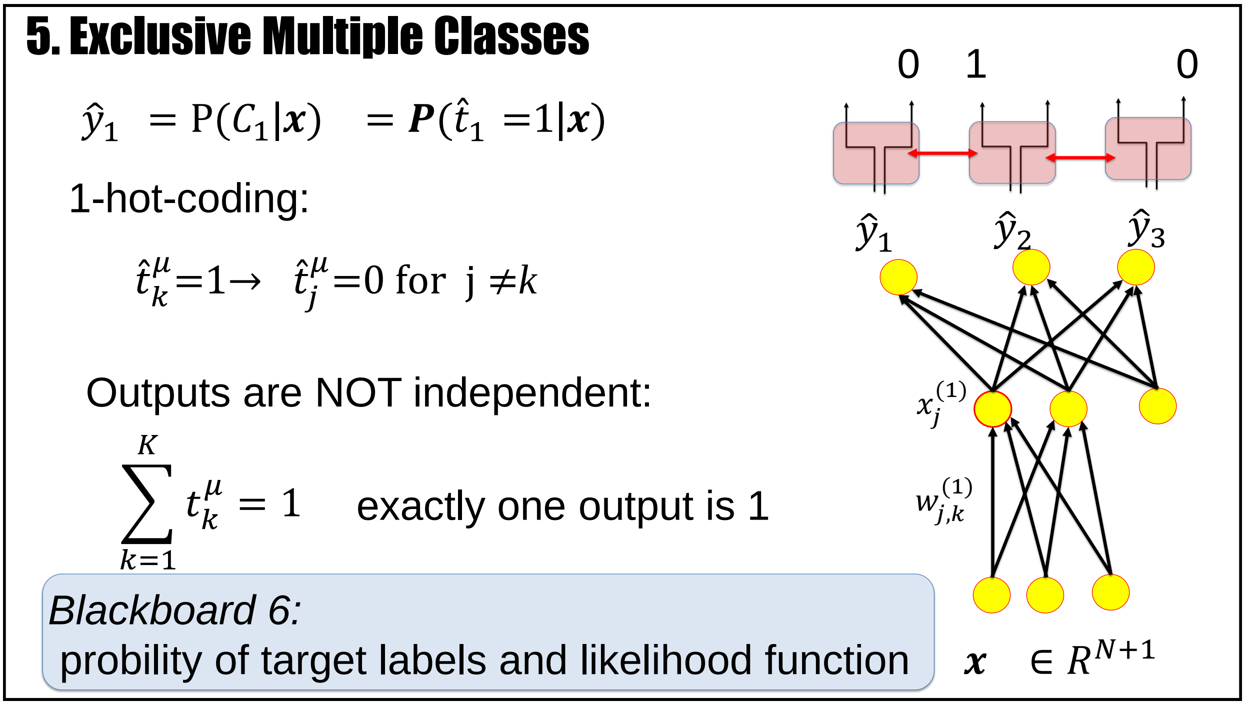
Mathematically speaking: if we want to interpret the output of neuron k as

$$\hat{y}_k = \mathrm{P}(C_k|\boldsymbol{x}) = \boldsymbol{P}(\hat{t}_k = 1|\boldsymbol{x})$$

then the outputs should interact with each other so that the output of neuron k is

$$\hat{y}_k = \frac{\boldsymbol{exp}(a_k)}{\sum_j \boldsymbol{exp}(a_j)}$$

The right-hand side is called the ‘softmax’ function



5. Exclusive Multiple Classes

$\hat{y}_1 = \mathrm{P}(C_1|\boldsymbol{x}) = \boldsymbol{P}(\hat{t}_1 = 1|\boldsymbol{x})$

1-hot-coding:

$\hat{t}_k^\mu = 1 \rightarrow \hat{t}_j^\mu = 0$ for j $\neq k$

Outputs are NOT independent:

$\sum_{k=1}^{K} t_k^\mu = 1$ exactly one output is 1

Blackboard 6:
probility of target labels and likelihood function

Previous slide.

For a single-class problem, we have seen that a maximum likelihood approach leads to a cross-entropy error function.

We repeat the derivation for the analogous situation of mutually exclusive classes.

Blackboard 6:
Probability of target labels: mutually exclusive classes

Your notes.

5. Cross entropy error for neural networks: Multiclass

We have a total of K classes (mutually exclusive: either dog or car)

Minimize* the **cross-entropy**

$$E(w) = -\sum_{k=1}^{K}\sum_{\mu}[t_k^\mu \, ln \, \hat{y}_k^\mu]$$

parameters= all weights, all layers

*Minimization under the constraint: $\sum_{k=1}^{K} \hat{y}_k^\mu = 1$

Compare: **KL divergence between outputs and targets**

$$\text{KL}(w) = -\{\textstyle\sum_{k=1}^{K}\sum_{\mu}[t_k^\mu ln \, \hat{y}_k^\mu] - \sum_{\mu}[t_k^\mu ln \, t_k^\mu]\}$$

$$\text{KL}(w) = E(w) + constant$$

Previous slide.

The cross-entropy error function for mutually exclusive classes is a generalization of the cross-entropy formula for a single class. To see this, assume that we have exactly two mutually exclusive classes: thus if the targets of the first output is 1 the target of the second output must be zero (and vice versa). This assumption leads back to the single-class cross-entropy formula.

For the multi-class problem, the cross-entropy error function can be interpreted as the KL divergence between actual outputs and targets plus a constant.
For the position of the minimum of the error function the constant is irrelevant.

Since output probabilities sum to one, and since we want to interpret the outputs as probabilities, the minimization must be performed under the constraint
$\sum_{k=1}^{K} \hat{y}_k^\mu = 1$.

Working with the softmax function in the output guarantees that this constraint is always satisfied

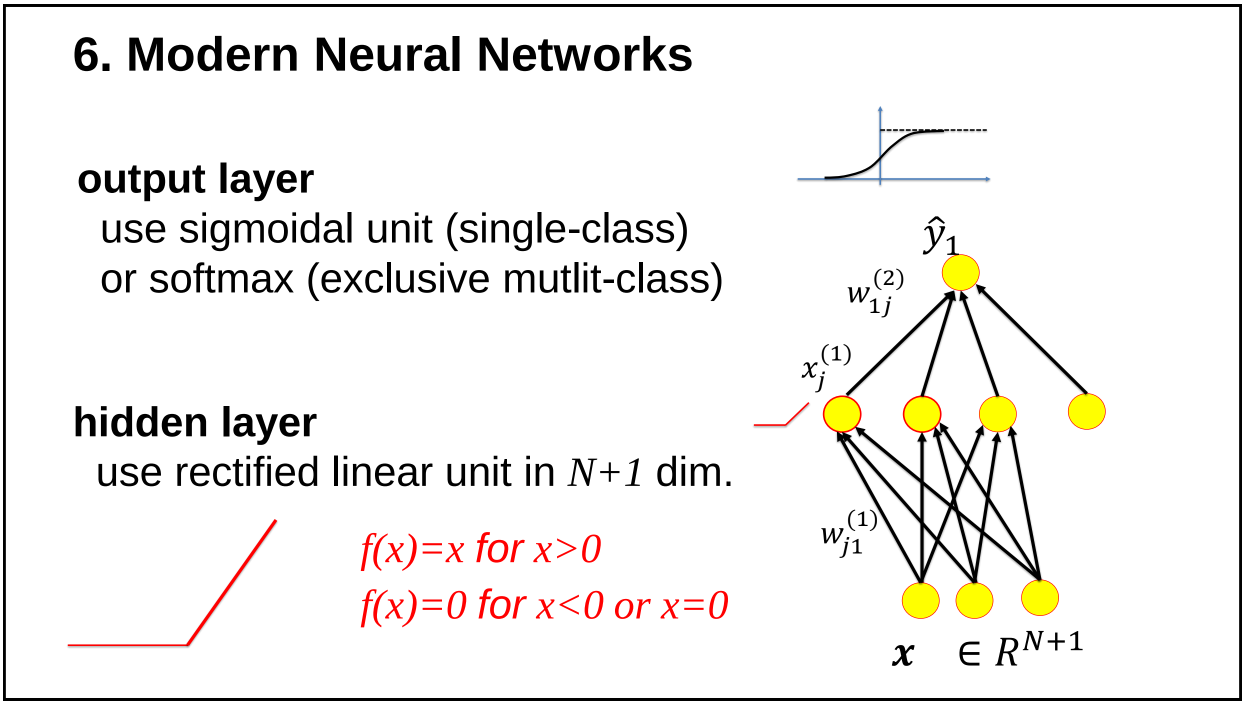
Artificial Neural Networks: Lecture 3
Statistical Classification by Deep Networks

Wulfram Gerstner
EPFL, Lausanne, Switzerland

1. The statistical view: generative model
2. The likelihood of data under a model
3. Application to artificial neural networks
4. Multi-class problems
5. Sigmoidal as a natural output function
6. **Rectified linear for hidden units**

Previous slide.

So far we only focused on the function used in the output layer. The question arises whether there is also an ‘ideal’ function for the hidden neurons.



6. Modern Neural Networks

output layer
use sigmoidal unit (single-class)
or softmax (exclusive mutlit-class)

hidden layer
use rectified linear unit in $N+1$ dim.

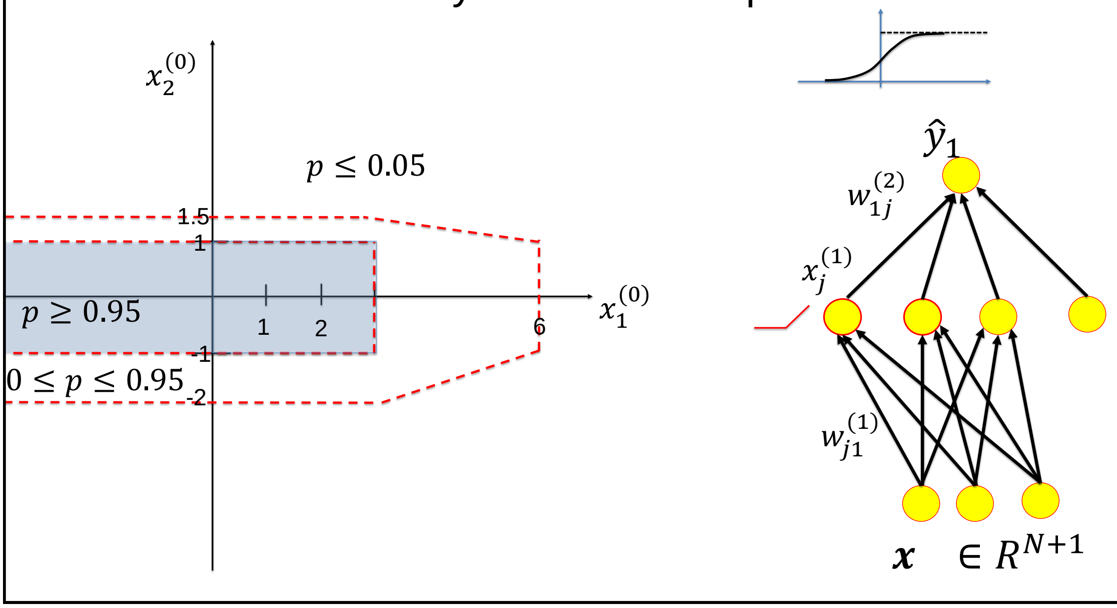
$f(x)=x$ for $x>0$
$f(x)=0$ for $x<0$ or $x=0$

$\hat{y}_1$
$w^{(2)}_{1j}$
$x^{(1)}_j$
$w^{(1)}_{j1}$
$\boldsymbol{x} \in R^{N+1}$

Previous slide.

There are no rigorous theoretical arguments. In practice, modern artificial neural networks often use piecewise linear units.

Preparation for Exercises: Link multilayer networks to probabilities



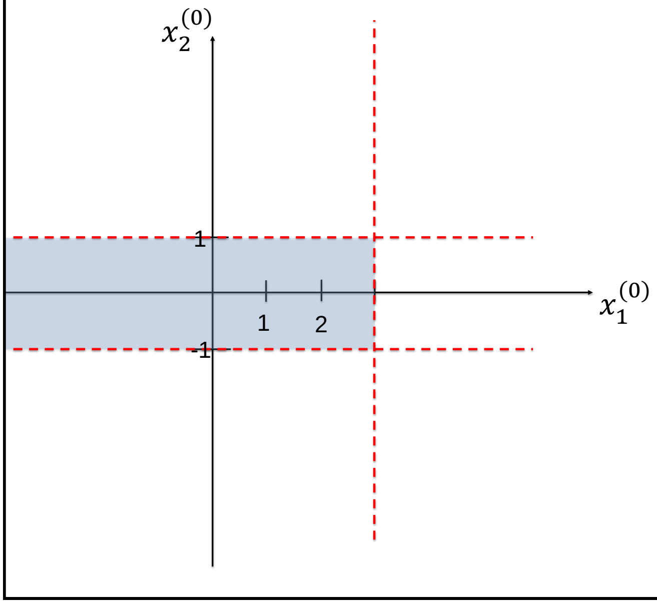
Previous slide.

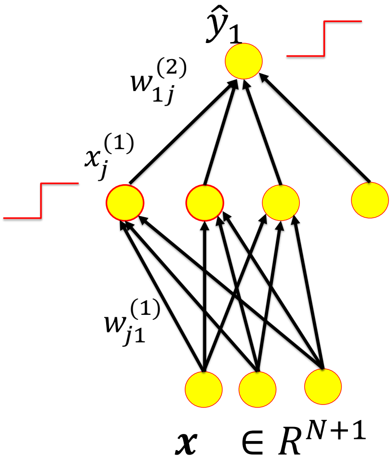
In the exercises, we will use a piecewise linear functions for the hidden units, and a single sigmoidal output function.

The blue shading indicates that at least 95 percent of the examples that lie inside the rectangular area belong to the class (and at most 5 percent in that area do not belong the class).
Outside the dashed red polygone, at least 95 percent of the examples do not belong to the class (and less than 5 percent belong to the class).

In order to construct the solution, we exploit the fact that for the sigmoidal output, a probability of p=0.95 corresponds to an activation of a=+3.

Preparation for Exercises: there are many solutions!!!!





Previous slide.

If we use step functions in the hidden layer, we cannot model probabilities. However, it is easier to develop an intuition about how the solutions could look like.

The blue shading on the left-hand side indicates that all positive examples lie inside the rectangular area.

QUIZ: Modern Neural Networks

[ ] piecewise linear units should be used in all layers

[ ] piecewise linear units should be used in the hidden layers

[ ] softmax unit should be used for exclusive multi-class in an output layer in problems with 1-hot coding

[ ] sigmoidal unit should be used for single-class problems

[ ] two-class problems (mutually exclusive) are the same as single-class problems

[ ] multiple-attribute-class problems are treated as multiple-single-class

[ ] In neural nets we can interpret the output as a probability,

$$\hat{y}_1 \;\; = \mathrm{P}(C_1|\boldsymbol{x})$$

[ ] if we are careful in the model design, we may interpret the output as a probability that the data belongs to the class

Artificial Neural Networks: Lecture 3

Statistical classification by deep networks

Wulfram Gerstner

EPFL, Lausanne, Switzerland

Objectives for today:

- The cross-entropy error is the optimal loss function for classification tasks
- The sigmoidal (softmax) is the optimal output unit for classification tasks
- Exclusive Multi-class problems use ‘1-hot coding’
- Under certain conditions we may interpret the output as a probability
- Piecewise linear units are preferable for hidden layers

Reading for this lecture:

Bishop 2006, Ch. 4.2 and 4.3

Pattern recognition and Machine Learning

or

Bishop 1995, Ch. 6.7 – 6.9

Neural networks for pattern recognition

or

Goodfellow et al.,2016 Ch. 5.5, 6.2, and 3.13 of

Deep Learning