

Artificial Neural Networks

Wulfram Gerstner

EPFL, Lausanne, Switzerland

Deep Nets4: Statistical classification by deep networks

Part 1: Questions and Aims of this Lecture: The statistical view

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

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

Review: Data base for Supervised learning (single output)

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

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

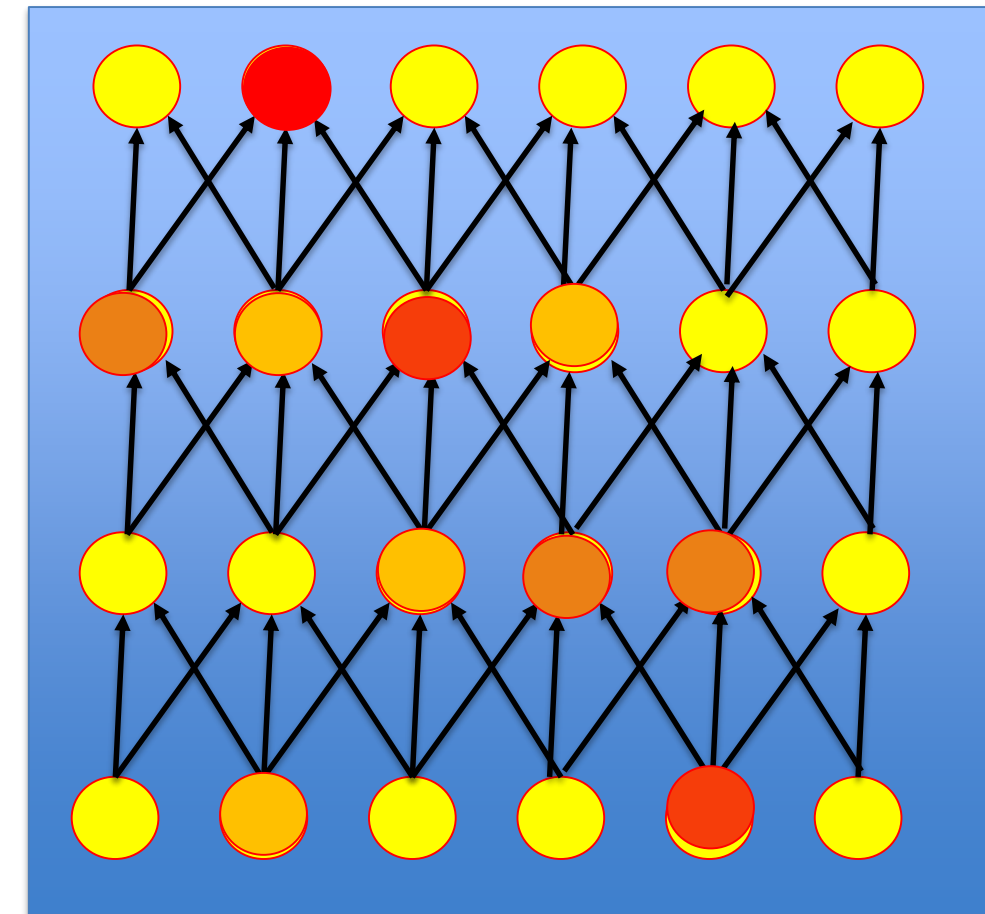
classifier

car (yes)

input



x^μ



Previous slide.

... and use the target information to adjust the parameters of our classifier which is a neural network in our case.

Review: Example MNIST



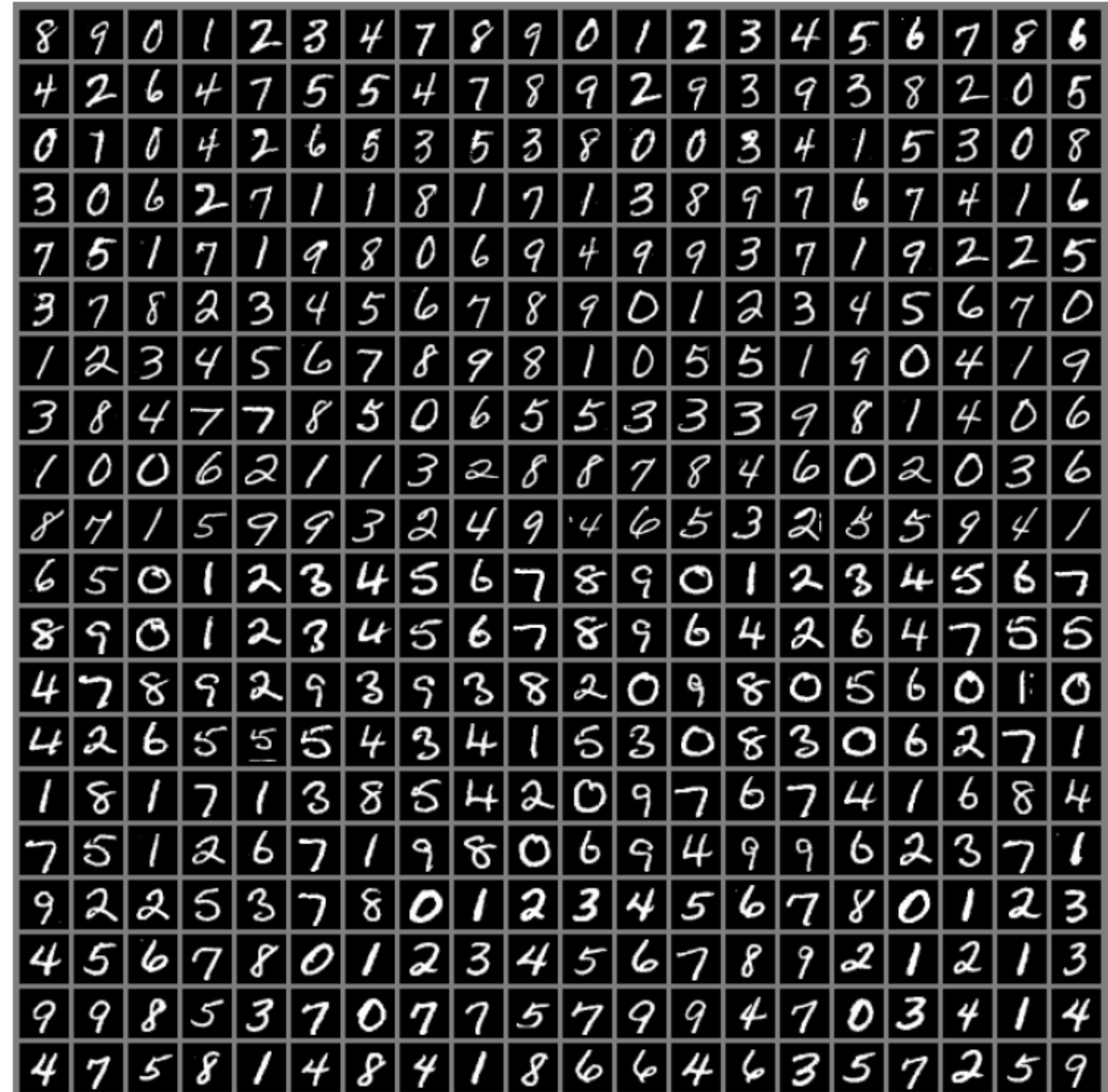
- images 28x28
- Labels: 0, ..., 9
- 250 writers
- 60 000 images in training set

Picture: Goodfellow et al, 2016

Data base:

<http://yann.lecun.com/exdb/mnist/>

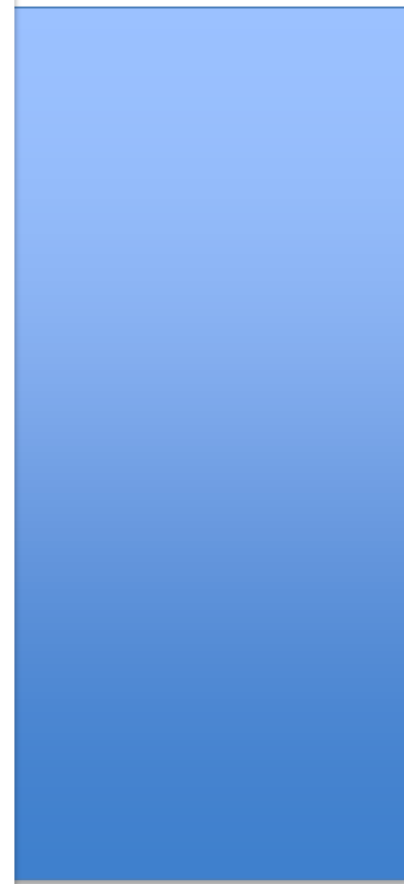
MNIST data samples



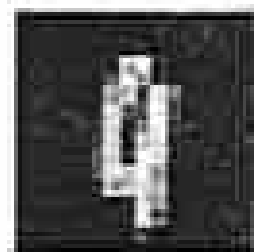
Previous slide.

As we have seen last week, all data is noisy

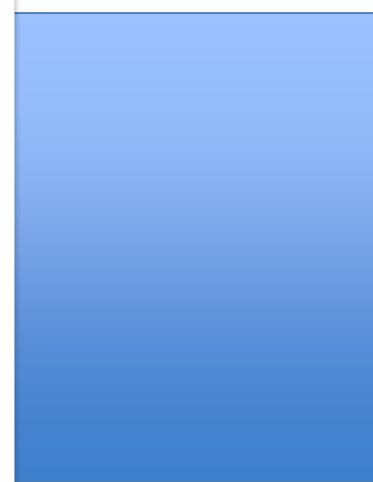
review: data base is noisy



9 or 4?



9 or 4?



- training data is always noisy
→ crossvalidation, regularization

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

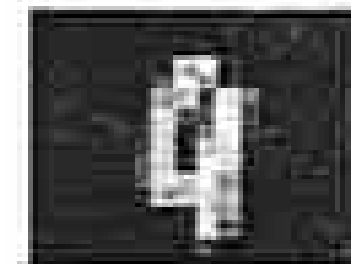
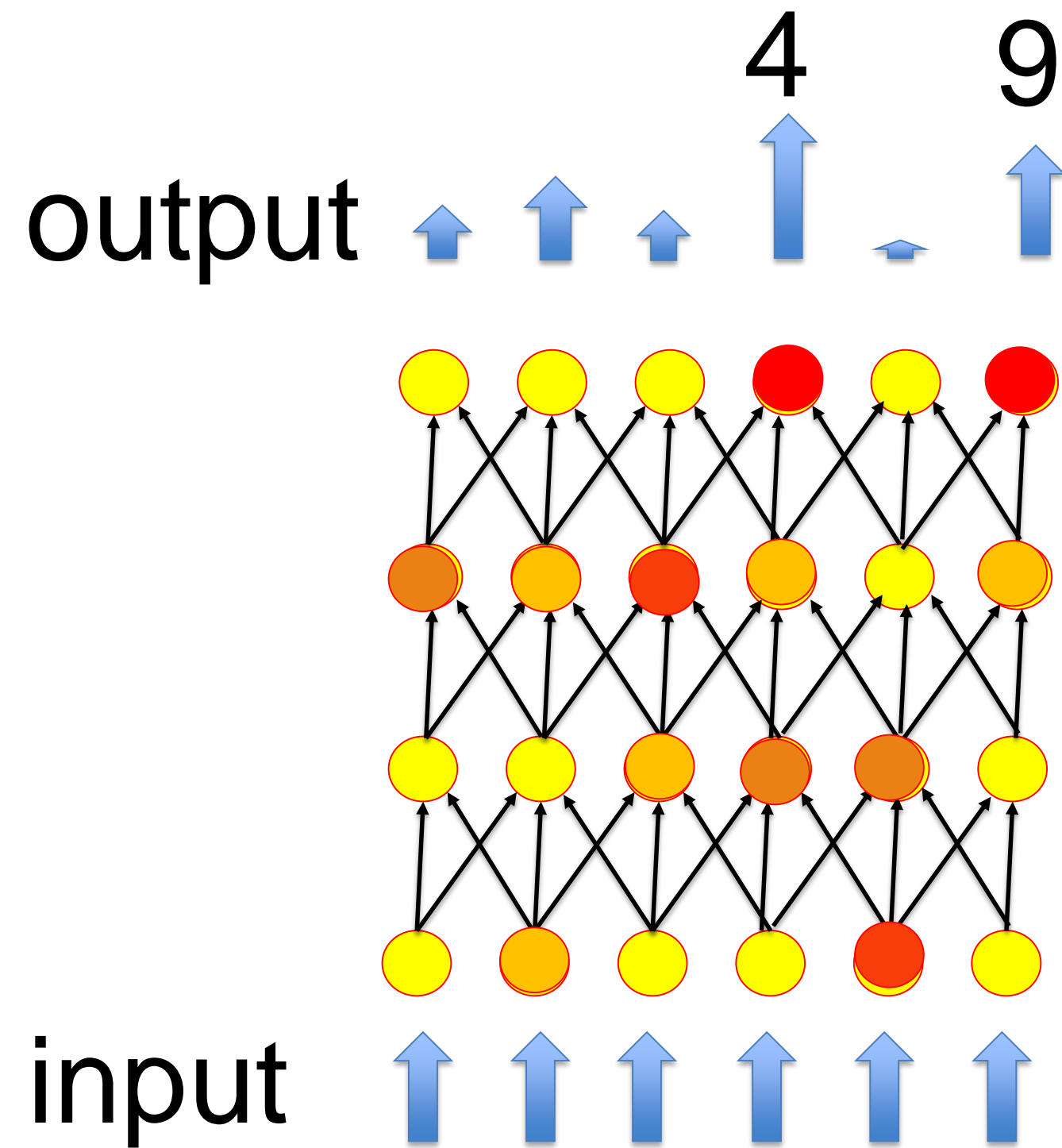
Previous slide.

Even the handwritten digits in the MNIST data base.

Note that the LABELS are noisy: what is a 4 for one writer is a 9 for another one.

Question for today

May we interpret the outputs of 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'?

The statistical view

A neural network should be seen as a
generative model
that predicts labels probabilistically

Previous slide.

A central notion for a probabilistic interpretation is the concept of a 'generative model'.

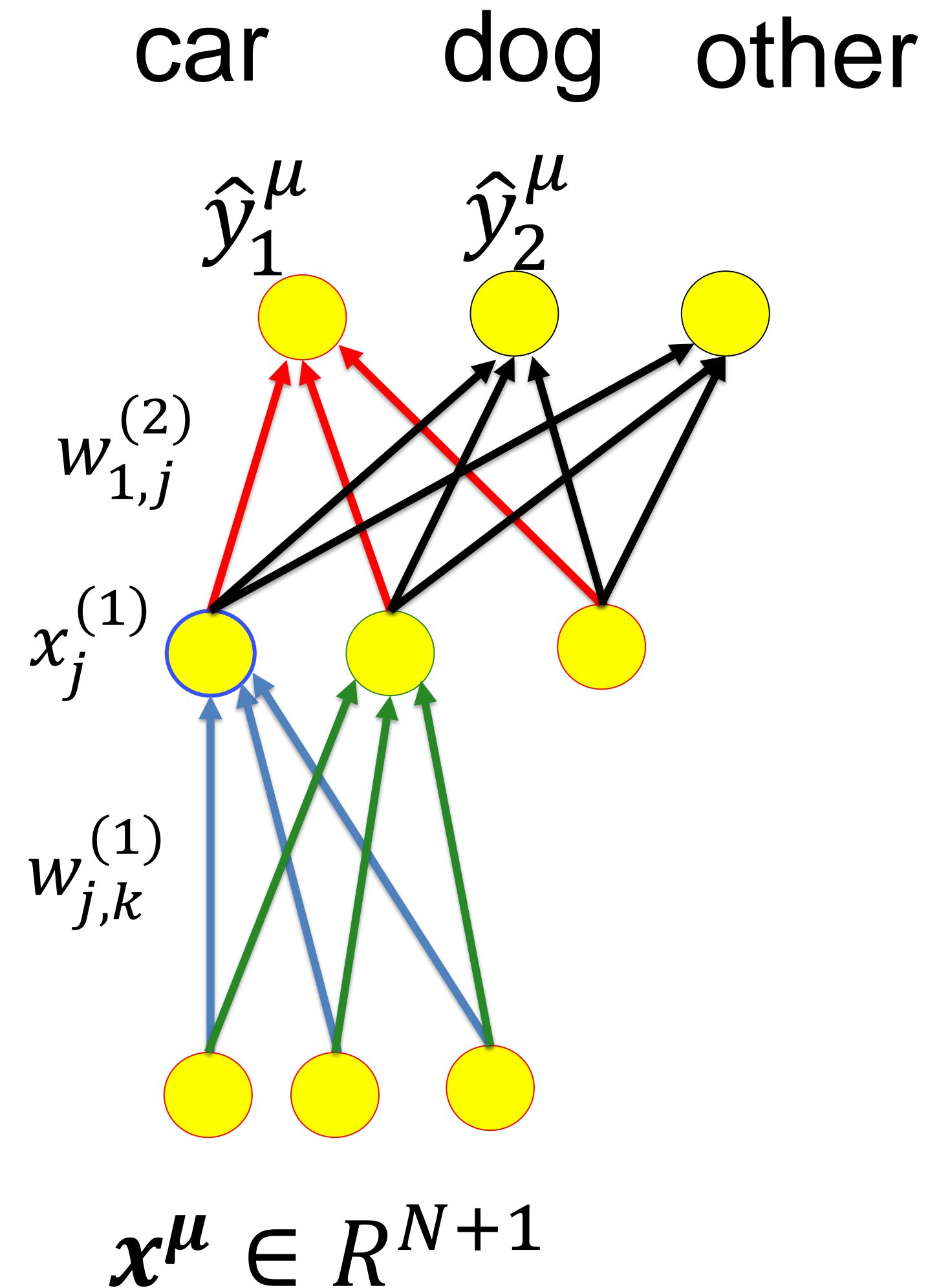
The statistical view

Idea:

interpret the output \hat{y}_k^μ
as the **probability** that
the input pattern \mathbf{x}^μ
should be classified
as class k

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

$$\hat{y}_k = P(C_k | \mathbf{x}) \quad \text{arbitrary novel pattern}$$



Previous slide.

The aim is to interpret the output of unit k

$$\hat{y}_k = P(C_k | \mathbf{x})$$

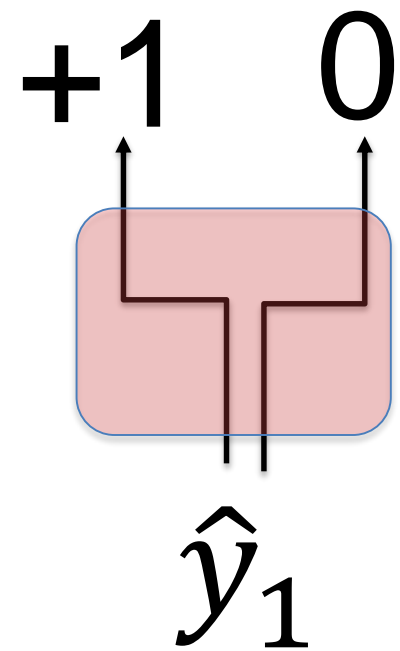
as the probability that the novel input pattern \mathbf{x} belongs to class C_k

The notation is that of conditional probabilities

$P(C_k | \mathbf{x})$ is the probability of class C_k given \mathbf{x} .

The statistical view: single class

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

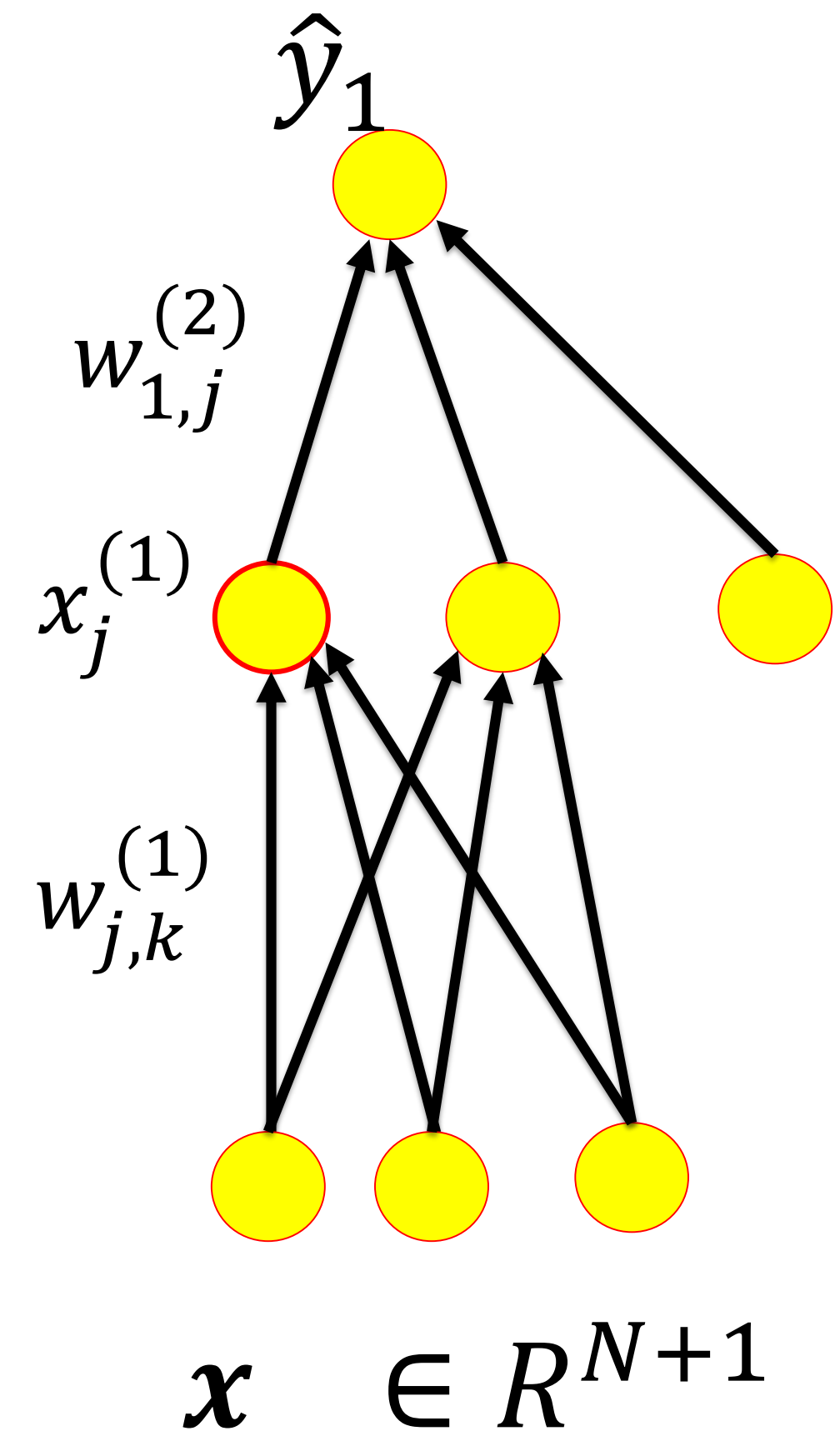


→ generative model for class label

$$\text{with } \hat{y}_1 = P(C_1|\mathbf{x}) = P(\hat{t}_1 = 1|\mathbf{x})$$

↑
predicted label

$$p_+ = \hat{y}_1 \quad \begin{array}{c} +1 \quad 0 \\ \uparrow \quad \uparrow \\ \text{[pink box]} \\ \downarrow \\ \hat{y}_1 \end{array} \quad p_- = 1 - \hat{y}_1$$



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.

Summary: the statistical view

Artificial Neural Network as generative model for class label

$$\text{with } \hat{y}_1 = P(C_1|\mathbf{x}) = P(\hat{t}_1 = 1|\mathbf{x})$$

↑
predicted label

→ basis for a statistical interpretation of neural networks

You work in the miniproject work with

- regularization methods
- ADAM optimizer
- ReLu for hidden units
- log-policy loss
- TD value and policy gradient

(Done, used in both miniprojects)

Done: Relevant for Miniproject on RL

- cross-entropy error function
- sigmoidal (softmax) output
- 1-hot coding for multiclass

This week: Relevant for Miniproject on image classification

Artificial Neural Networks

Wulfram Gerstner

EPFL, Lausanne, Switzerland

Statistical Classification by Deep Networks

Part 2: The likelihood of data under a model

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.

The likelihood of a model (given data)

Overall aim:

What is the probability that my set of P data points

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

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?

The likelihood of a model (Repetition from ML class)

Detour:

forget about labeled data, and just think of input patterns

What is the probability that a set of P data points

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

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. This part is a repetition of topics that have already covered in other classes such as the Machine Learning class of Jaggi and Urbanke.

Example: Gaussian distribution

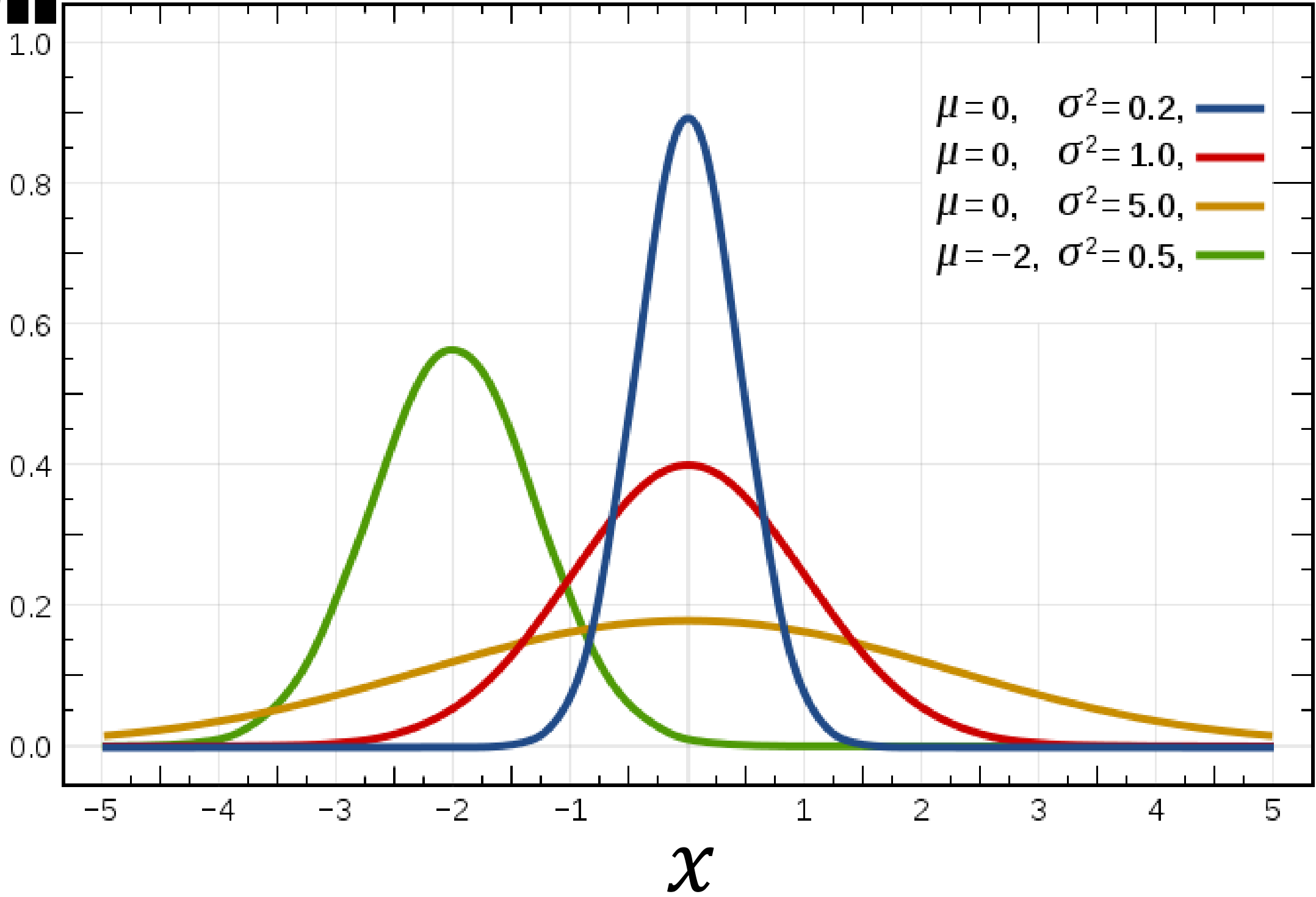
$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$$

this depends on 2 parameters

$$\{w_1, w_2\} = \{\mu, \sigma\}$$

center

width



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

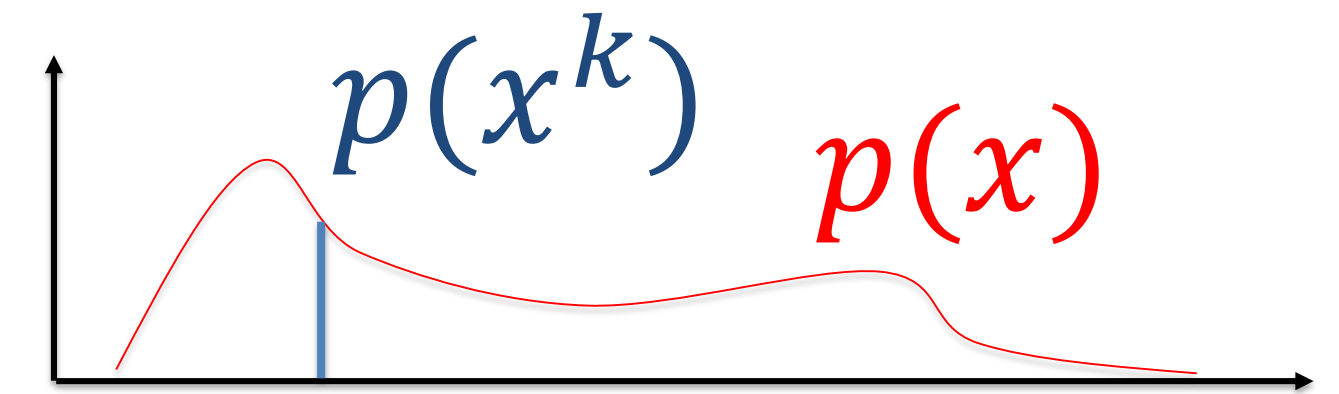
Previous slide.

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.

Random Data Generation Process

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

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



What is the probability to generate P data points?

$$P = p(x^1)\Delta x \quad p(x^2)\Delta x \quad \dots \quad p(x^k)\Delta x \quad \dots \quad p(x^P)\Delta x$$

$$P = \prod_{n=1}^P [p(x^n)\Delta x] = \prod_{n=1}^P [p(x^n)] \cdot [\Delta x]^P$$

$$P = p_{model}(X) \cdot C$$

Product of two terms: C does not depend on the choice of model

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

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

is

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

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

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.

Because of (assumed) independence of the data generation process, we can just multiply the probabilities of all data points.

Probability to generate P data points

$$P = p_{model}(X) \cdot C$$

Here $p_{model}(X)$ is the '**likelihood**' that the specific set of P data points is generated by the model.

Note that C does not depend on the choice of model (but only on the discretization Δx)

Previous slide.

For optimization only the part that depends on the model (or the model parameters) is important. Therefore the constant C can be dropped.

Hence, $p_{model}(\mathbf{X})$ is not a 'probability' but some quantity 'proportional' to the probability. I call it likelihood (in a broad sense).

Likelihood function

Suppose the probability for generating a data point x^k using my model is proportional to

$$p(x^k)$$

Suppose that data points are generated independently.

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

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

could have been generated by my model is

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

Previous slide.

Note that we performed a small but important change in perspective.

We no longer generate data points, but we ask whether the observed data points **COULD HAVE BEEN GENERATED BY MY MODEL.**

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

$$p(x^1) p(x^2) p(x^3) \dots p(x^P)$$

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.

Example: for the specific case of the Gaussian

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

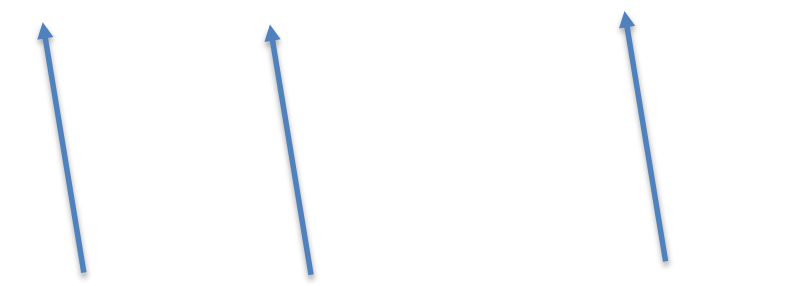
$$p_{model}(\mathbf{X}) = \left[\frac{1}{\sqrt{2\pi}\sigma} \right]^P \exp \left\{ \frac{-\sum_{k=1}^P (x^k - \mu)^2}{2\sigma^2} \right\}$$

Maximum Likelihood

$$p_{model}(\mathbf{X}) = p(\mathbf{x}^1) p(\mathbf{x}^2) p(\mathbf{x}^3) \dots p(\mathbf{x}^P)$$

BUT this likelihood depends on the parameters of my model

$$p_{model}(\mathbf{X}) = p_{model}(\mathbf{X} | \{w_1, w_2, \dots, w_n\})$$


parameters

Choose the parameters such that the likelihood is maximal!

Previous slide.

The expression $p_{model}(\mathbf{X}|\{w_1, w_2, \dots, w_n\})$

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.

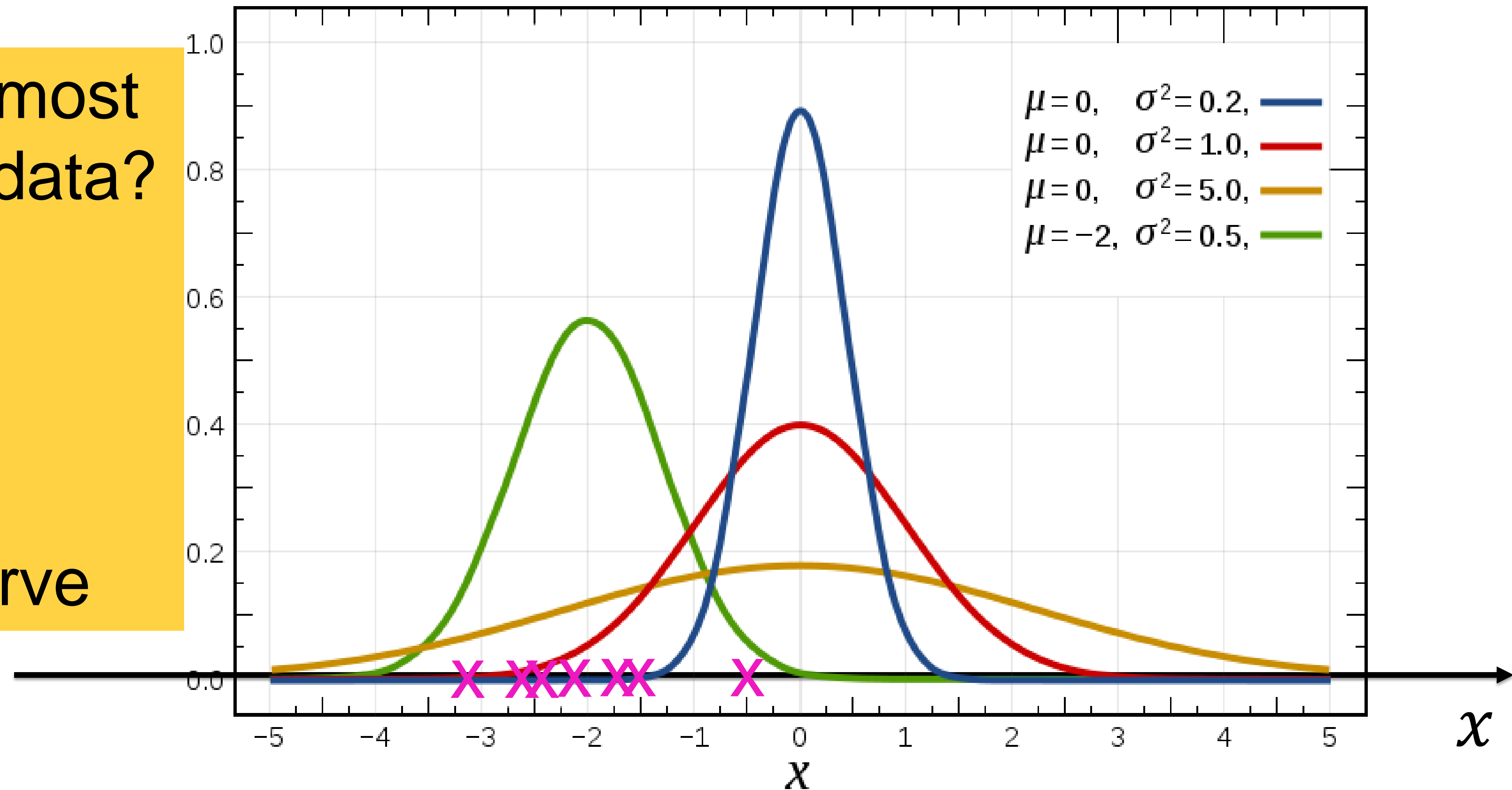
This optimization procedure is called the Maximum-Likelihood (ML) approach.

Example: Gaussian distribution

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

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.

Example: Gaussian (Repetition from ML class)

You have p data points.

$$p_{model}(\mathbf{X}) = p(\mathbf{x}^1) p(\mathbf{x}^2) p(\mathbf{x}^3) \dots p(\mathbf{x}^P)$$

The likelihood depends on the 2 parameters of the Gaussian

$$p_{model}(\mathbf{X}) = p_{model}(\mathbf{X} | \{w_1, w_2\})$$

$$p_{model}(\mathbf{X}) = p_{model}(\mathbf{X} | \{x^{center}, \sigma\})$$

Maximize $p_{model}(\mathbf{X})$ with respect to the model parameters

Steps:

- 1) You take the general likelihood formula and insert for each $p(x)$ the Gaussian function.
- 2) The likelihood depends on the choice of the parameters of the Gaussian.
- 3) You maximize the likelihood with respect to the parameter x^{center} that represents the center of the Gaussian; to do so, set the derivative to zero.
- 4) The result is

$$x^{center} = \frac{1}{P} \sum_{k=1}^P x^k$$

See: Solution of exercise 1.

Also this whole section is also treated in the class:
Introduction to Machine Learning

Gaussian: best parameter choice for center

$$x^{center} = \frac{1}{P} \sum_{k=1}^P x^k$$

Conclusion:

choosing parameters such that they maximize the likelihood that the data **COULD HAVE BEEN GENERATED** by a Gaussian, yields a reasonable parameter choice.

NOTE: we never assume that the data was actually generated by a Gaussian.

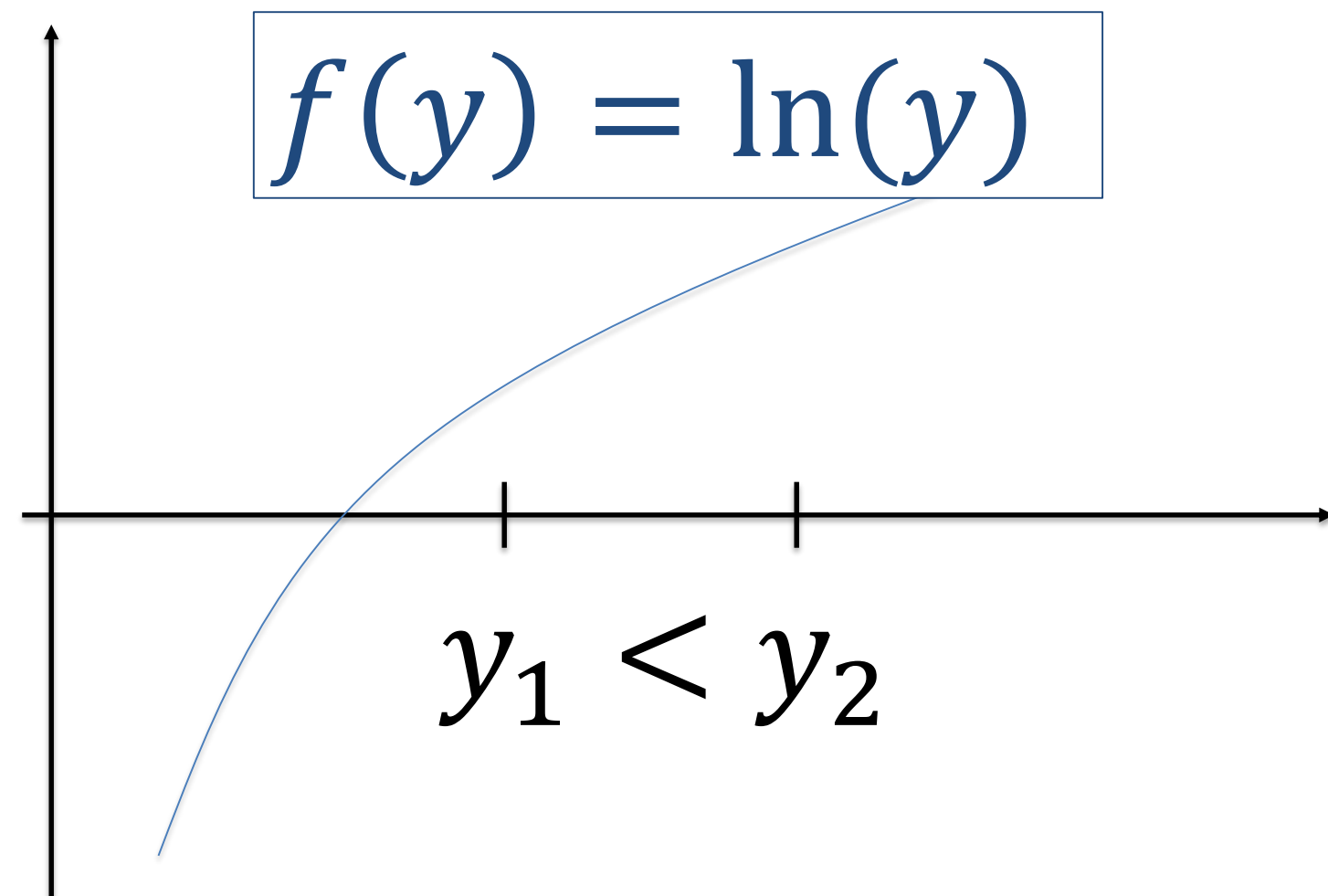
Your notes.

Maximum Likelihood (general)

Choose the parameters such that the likelihood

$$p_{model}(\mathbf{X} | \{w_1, w_2, \dots, w_n\}) = p(\mathbf{x}^1) p(\mathbf{x}^2) p(\mathbf{x}^3) \dots p(\mathbf{x}^P)$$

is maximal



Note:

Instead of maximizing

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

you can also maximize

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

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

$$p_{model}(\mathbf{X}|\{w_1, w_2, \dots, w_n\})$$

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

$$\ln[p_{model}(\mathbf{X}|\{w_1, w_2, \dots, w_n\})]$$

Maximum Likelihood (general)

Choosing the parameters such that the likelihood

$$p_{model}(\mathbf{X} | \{w_1, w_2, \dots, w_n\}) = p(x^1) p(x^2) p(x^3) \dots p(x^P)$$

is maximal is equivalent to maximizing the log-likelihood

$$LL(\{w_1, w_2, \dots, w_n\}) = \ln(p_{model}) = \sum_k \ln p(x^k)$$

“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.

Similarly, the observed data was certainly not generated by a neural network. Nevertheless you can fit it by a neural network, using the maximum likelihood approach.

Summary: Maximum Likelihood Method

Choose the parameters such that the likelihood

$$p_{model}(X|\{w_1, w_2, \dots, w_n\}) = p(x^1) p(x^2) p(x^3) \dots p(x^P)$$

is maximal.

These parameters are found by maximizing the log-likelihood

$$LL(\{w_1, w_2, \dots, w_n\}) = \ln(p_{model}) = \sum_k \ln p(x^k)$$

Note 1): 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'.

Note 2): We can consider $-LL(\{w_1, w_2, \dots, w_n\})$ as a 'loss function'.

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'.

Notes:

- 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.
- Even though I called this section: 'the likelihood of data under a given model', some textbooks actually use the term of likelihood only in the sense of 'the likelihood of a model given the data'. In my opinion both views are OK. The more restrictive, second point of view emphasizes that you have to optimize the model parameters so as to maximize the likelihood.
- The negative log-likelihood is a standard loss function in statistics.

Artificial Neural Networks

Wulfram Gerstner

EPFL, Lausanne, Switzerland

Statistical Classification by Deep Networks

Part 3: Statistical interpretation of artificial neural networks

1. The statistical view: generative model
2. The likelihood of data under a model
- 3. Statistical interpretation of artificial neural networks**
 - 3A The cross-entropy loss function**

Previous slide.

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

The likelihood of data under a neural network model

Overall aim:

What is the likelihood that my set of P data points

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

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?

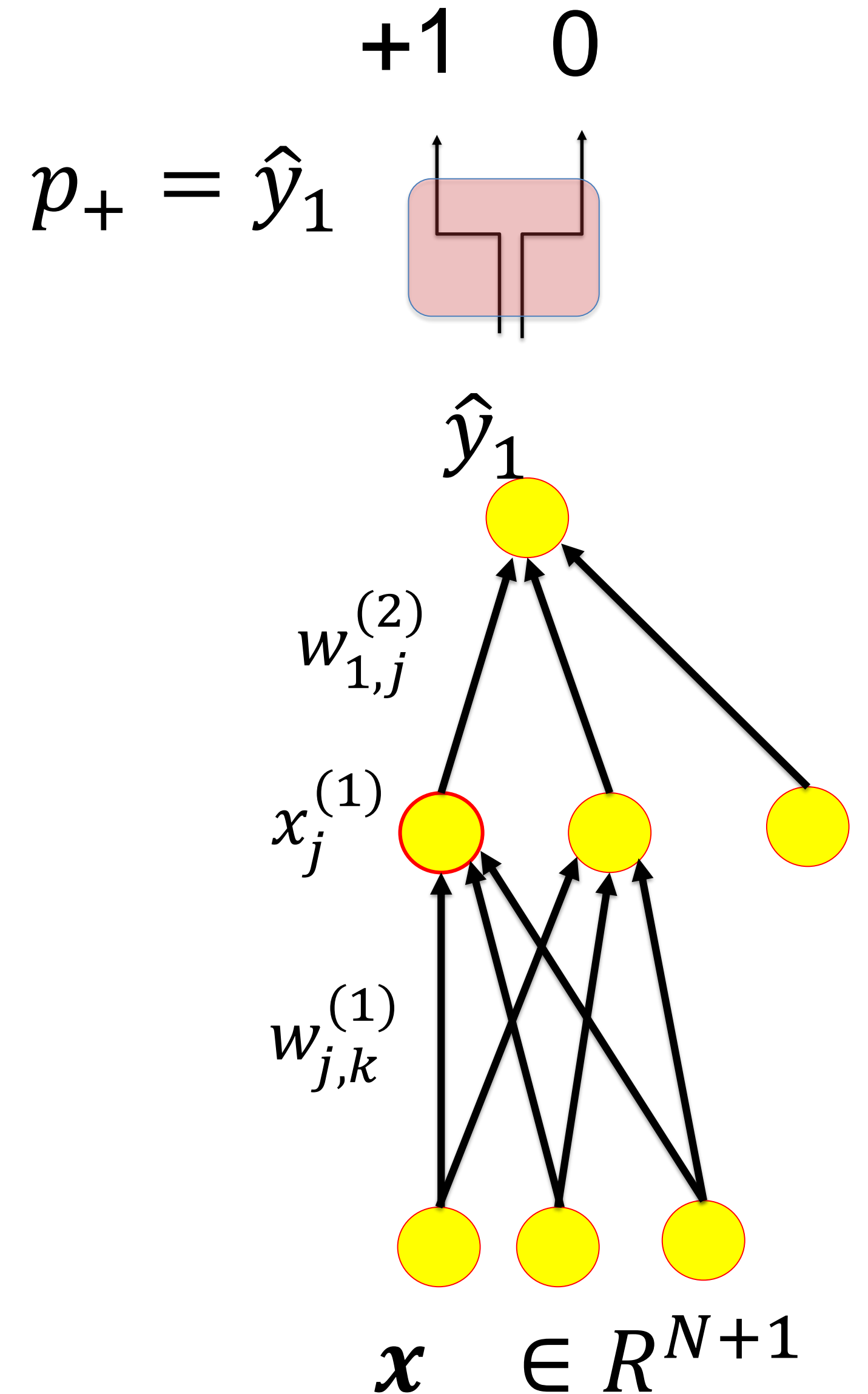
Likelihood of input-output pairs

generation of predicted labels

network output

study point \mathbf{x}^μ with $t^\mu=+1$: Probability that $(\mathbf{x}^\mu, +1)$ could have been generated

study point \mathbf{x}^μ with $t^\mu=0$: Probability that $(\mathbf{x}^\mu, 0)$ could have been generated



Previous slide.

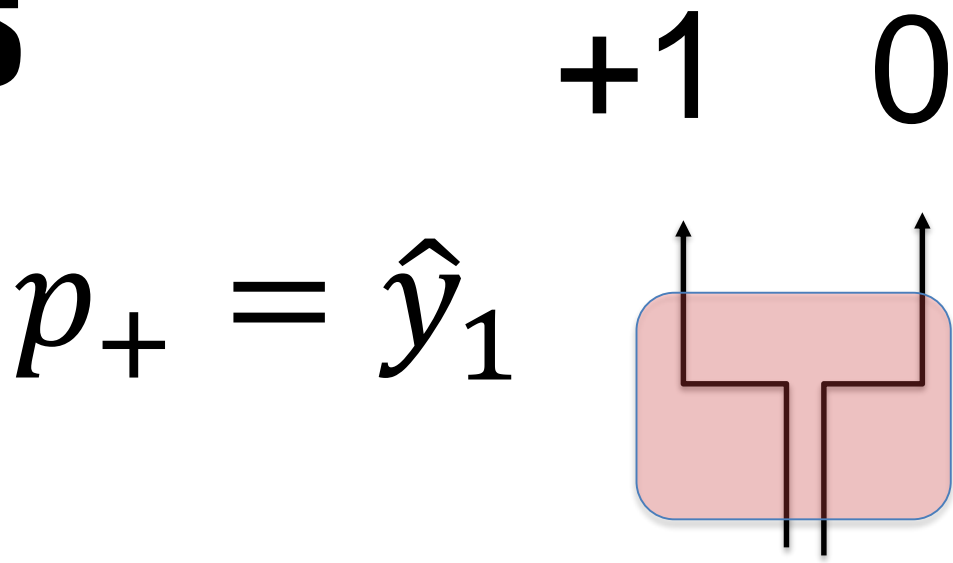
To analyze this we return to the generative model where the neural network output is interpreted as the probability to generate a label 1 or 0.

Likelihood of P input-output pairs

Your notes.

Maximum Likelihood for neural networks

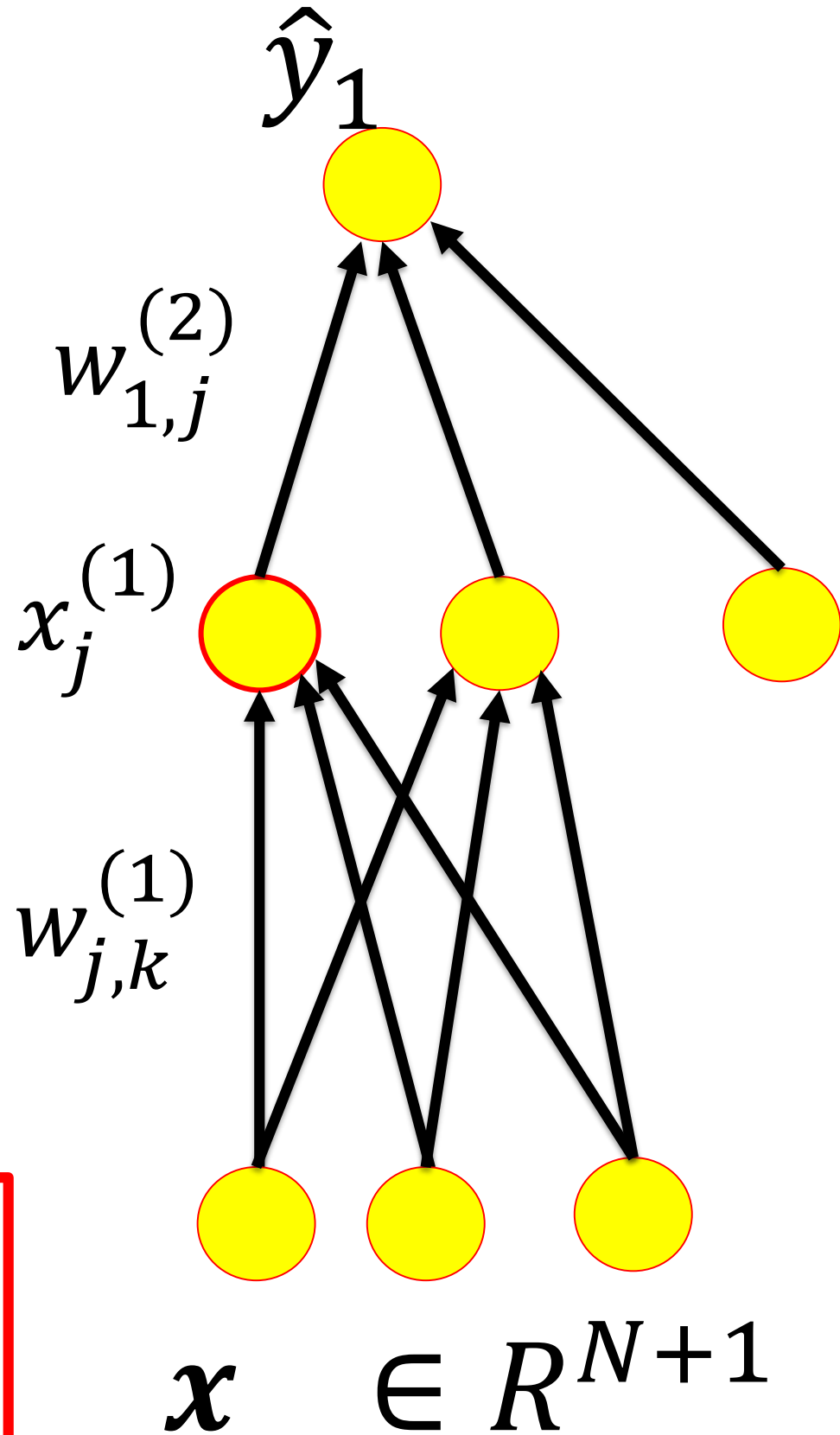
Minimize the negative log-likelihood



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

parameters= all weights, all layers

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



For single-class classification tasks, the cross-entropy loss is the 'natural' error function.

Previous slide.

If we consider the output as a model of the probability that a given data point x belongs to the class C , then a maximum likelihood approach implies that the correct error function is the cross-entropy loss.

Hence, for classification tasks we should not work with the quadratic loss, but with the cross-entropy loss.

Artificial Neural Networks

Wulfram Gerstner

EPFL, Lausanne, Switzerland

Statistical Classification by Deep Networks

Part 3: Statistical interpretation of artificial neural networks

1. The statistical view: generative model
 2. The likelihood of data under a model
 3. **Statistical interpretation of artificial neural networks**
- 3B Can we interpret the output as probability?**

Previous slide.

The central question is: can we REALLY interpret the output as a probability?

Cross-entropy error function for neural networks

Suppose we minimize the cross-entropy error function

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

Can we be sure that the output \hat{y}^{μ} 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.

Compared to the previous section, we now 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.

Output = probability ?

Suppose we minimize the cross-entropy error function

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

Assume

A We have enough data for training

B We have a sufficiently flexible network

Aim:

From Cross-entropy to output probabilities

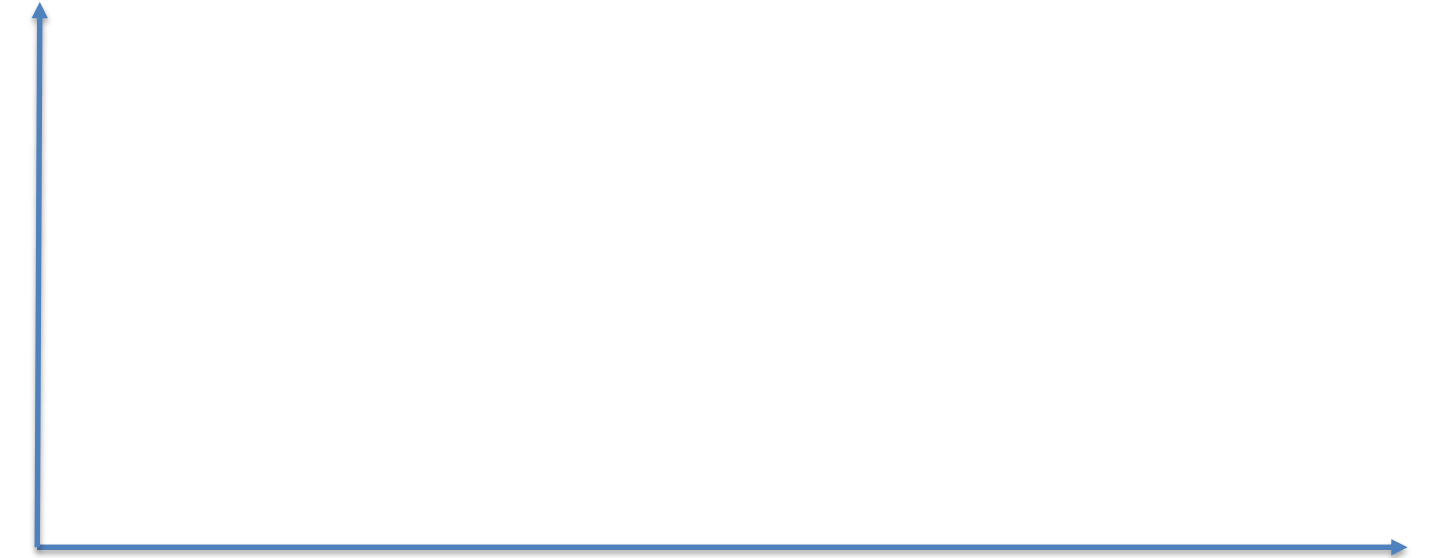
Your notes.

From Cross-entropy to output probabilities

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

Hypothesis A: we have many examples

Hypothesis B: network is flexible enough



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.

Summary: From Cross-entropy loss to probabilities

Hypothesis A: we have many examples

Hypothesis B: network is flexible enough

Then minimizing the cross-entropy error function

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

leads to an output

$$\hat{y} \approx \frac{n_1(x)}{n_0(x) + n_1(x)} \approx P(C|\mathbf{x})$$

Note: - flexibility at x must somehow 'match' the number of examples at x .
- in high dimensions we require LOTS of data!

QUIZ: Maximum likelihood solution means

- find the unique set of parameters that generated the data
- find the set of parameters that best explains the data
- find the best set of parameters such that your model could have generated the data

Minimization 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 $\hat{y}_1 = P(C_1|\mathbf{x})$
- guarantees that the output \hat{y}_1 of your network can be interpreted as $\hat{y}_1 = P(C_1|\mathbf{x})$

Artificial Neural Networks

Statistical Classification by Deep Networks

Part 4: Sigmoidal as a natural output function

1. The statistical view: generative model
2. The likelihood of data under a model
3. Statistical interpretation of 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.

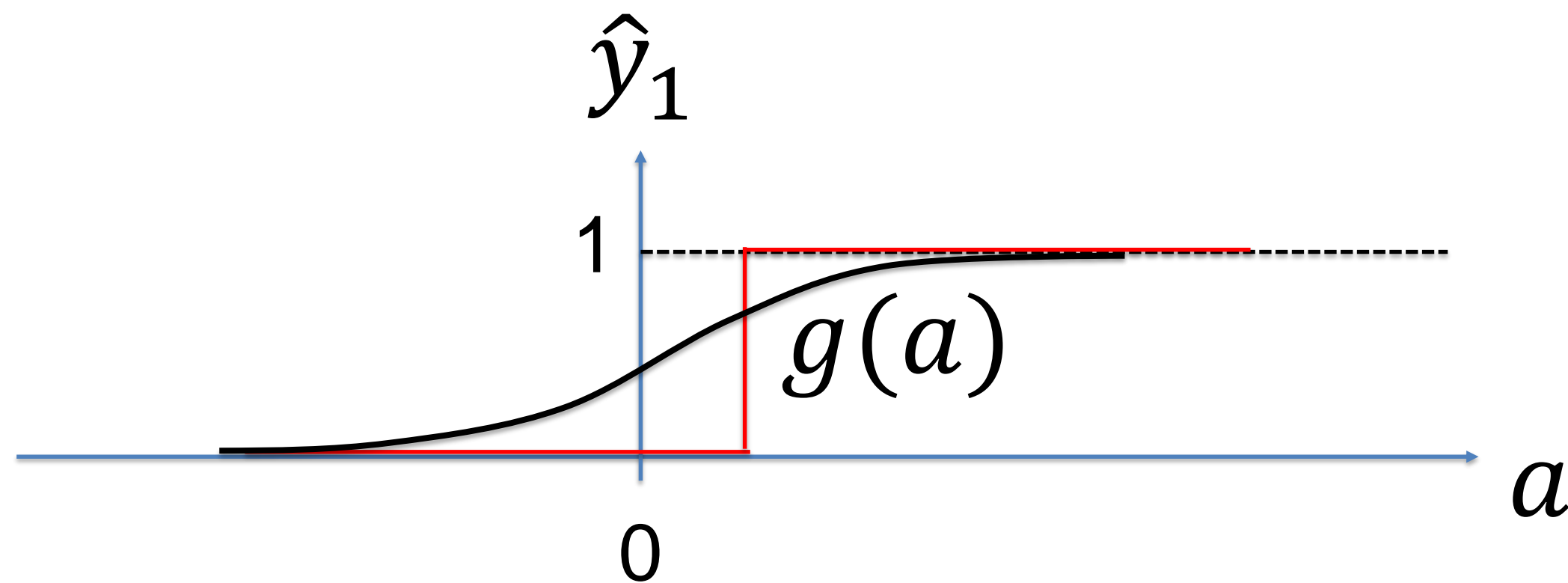
Why sigmoidal output ? – single class

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

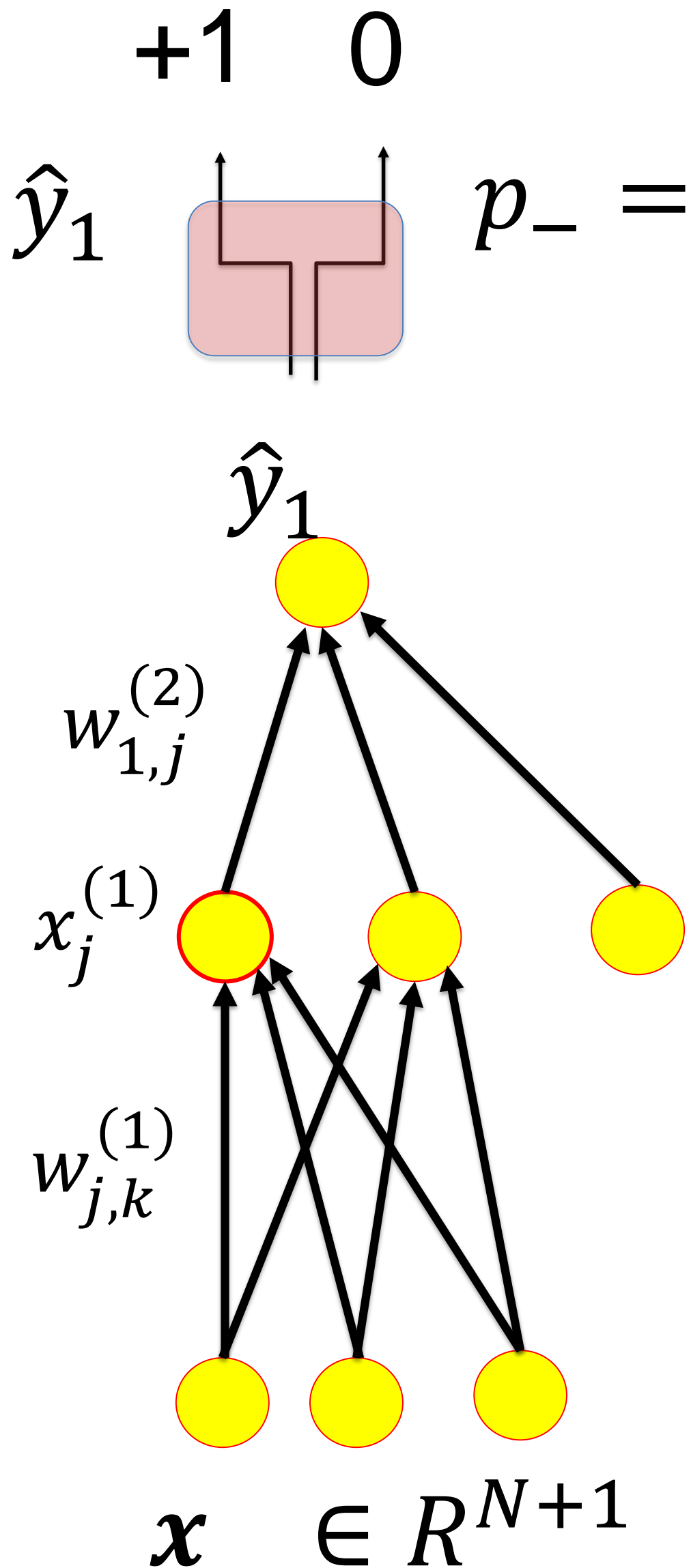
$$p_+ = \hat{y}_1 \quad p_- = 1 - \hat{y}_1$$

Observations (single-class):

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



natural choice/best choice for $g(a)$?



Previous slide.

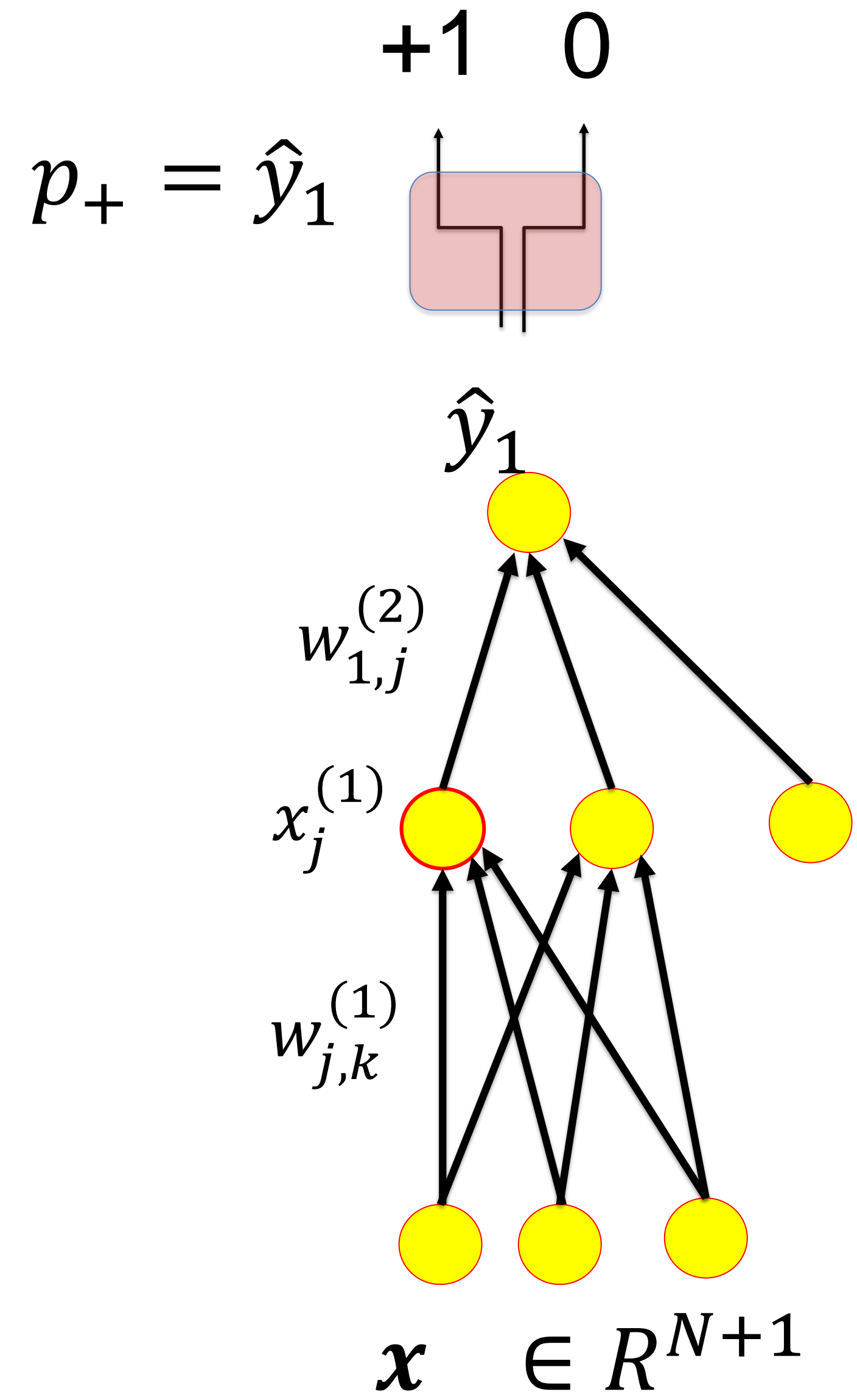
Suppose that we have taken all the necessary steps to interpret the output as a probability. What does this imply for the shape of the output function $g(a)$?

If the output is a probability, then it should lie in the range $[0,1]$. Thus the output must be bounded from above and from below. Moreover, a smooth function (black) will be better suited to express probabilities than a step function (red).

We now derive a 'natural' shape of the output function $g(a)$.

Note that I avoid the term 'optimal' shape because the derivation shows that other choices would be possible – but the sigmoidal function will come out as something very convenient.

derive optimal sigmoidal



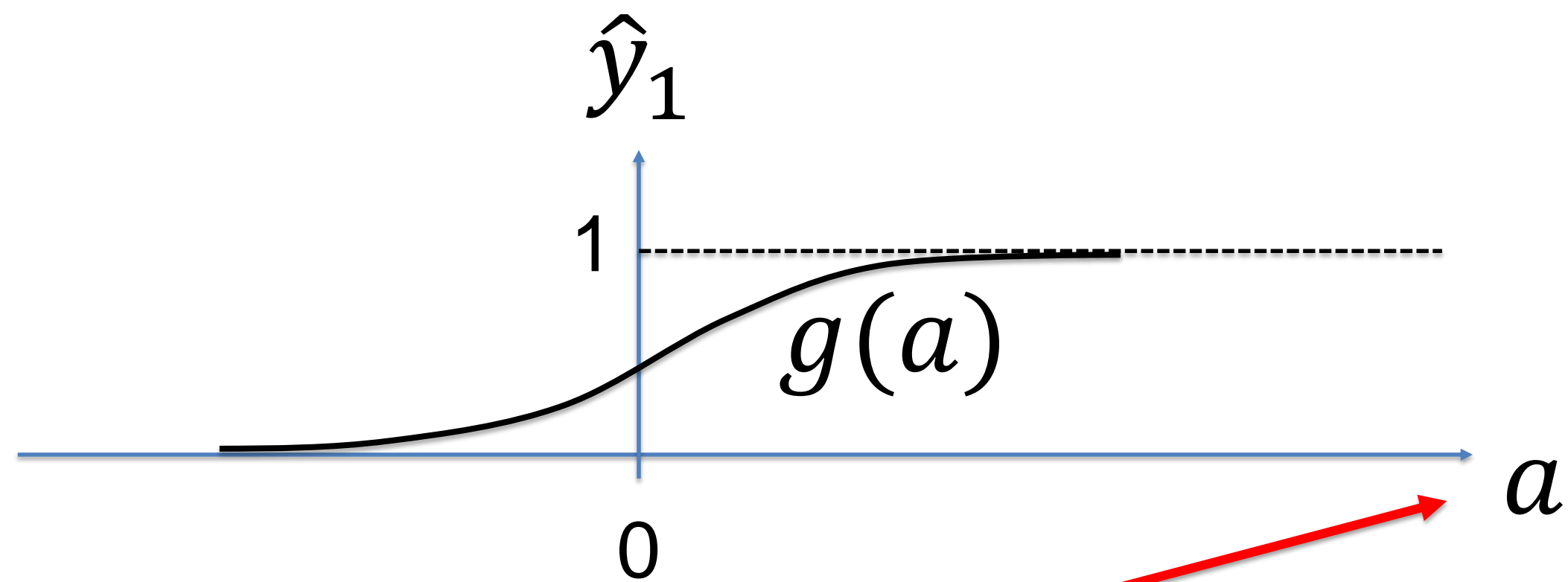
Your notes.

Why sigmoidal output ? – single class

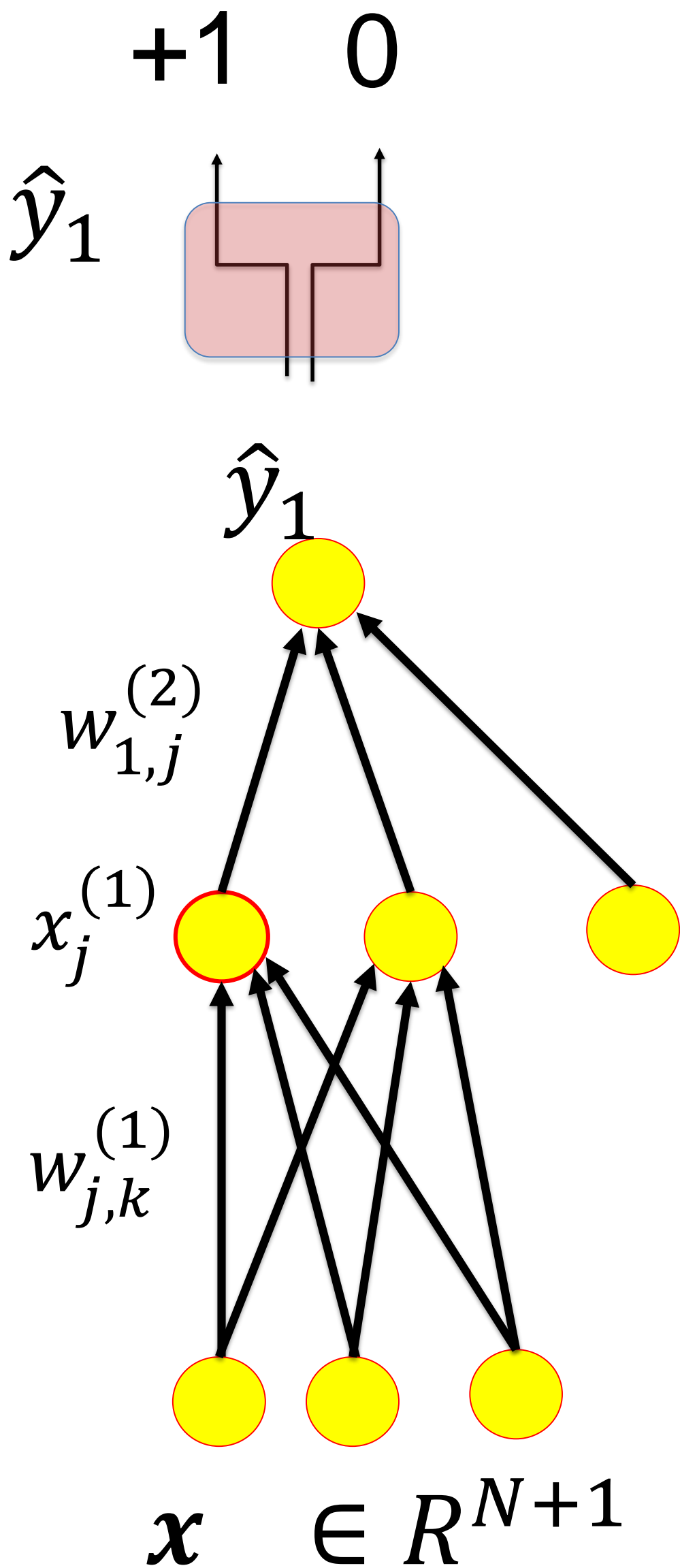
$$\hat{y}_1 = P(C_1|\mathbf{x}) = P(\hat{t}_1 = 1|\mathbf{x})$$

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

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



total input a (“activation/drive”) of output neuron can be interpreted as log-prob. ratio



$$\mathbf{x} \in R^{N+1}$$

Previous slide.

In a probabilistic setting, a natural choice for the output function is

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

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

probability $p(x,C)$ that the input is x and belongs to the class

\ln -----

probability $p(x, \setminus C)$ that the input is x and does not belong to the class

called the log-probability ratio. Here $\setminus C$ 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 $\ln[p(C|x)/p(\setminus C|x)]$

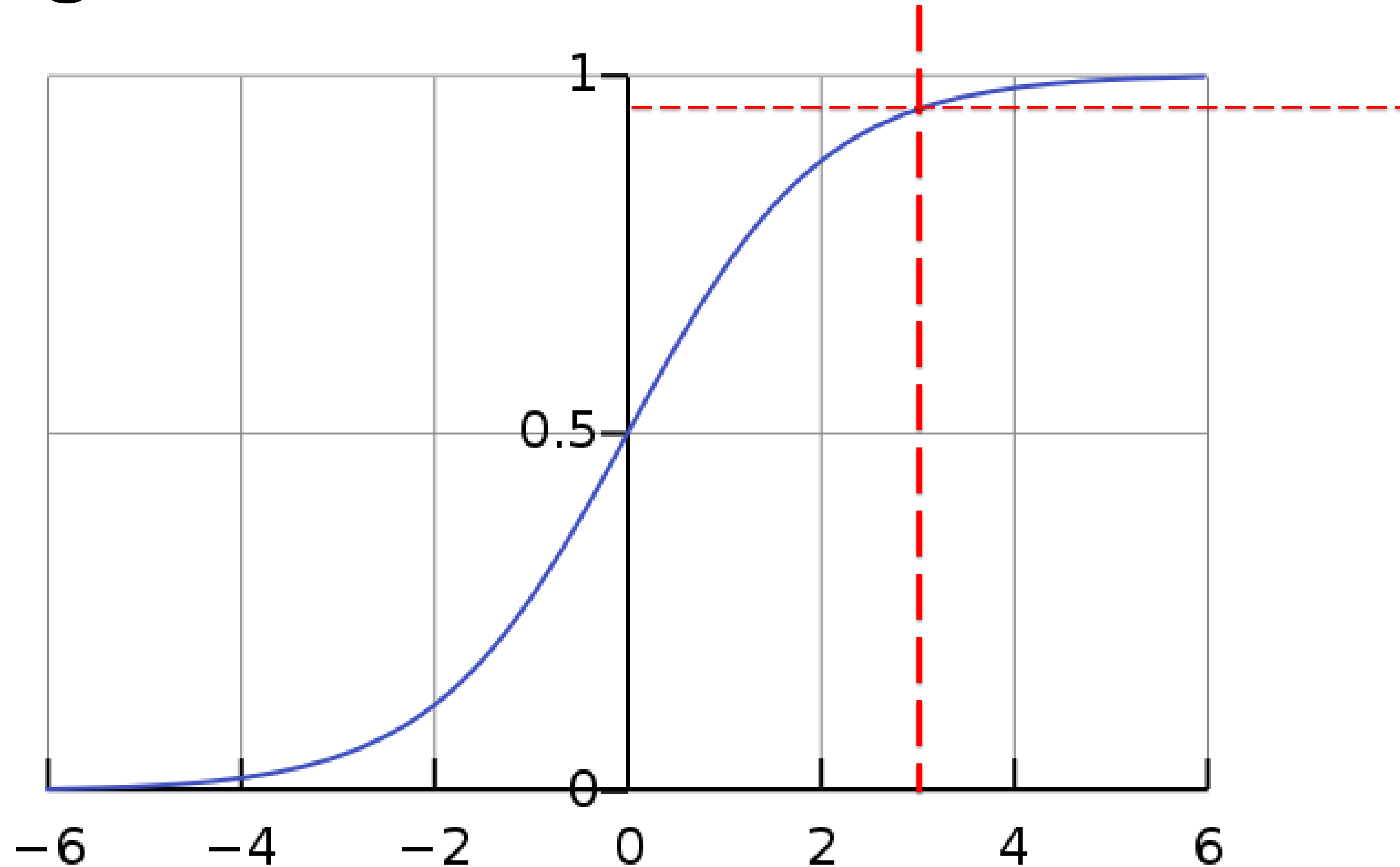
sigmoidal output = logistic function

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

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.

Summary: Why sigmoidal output ?

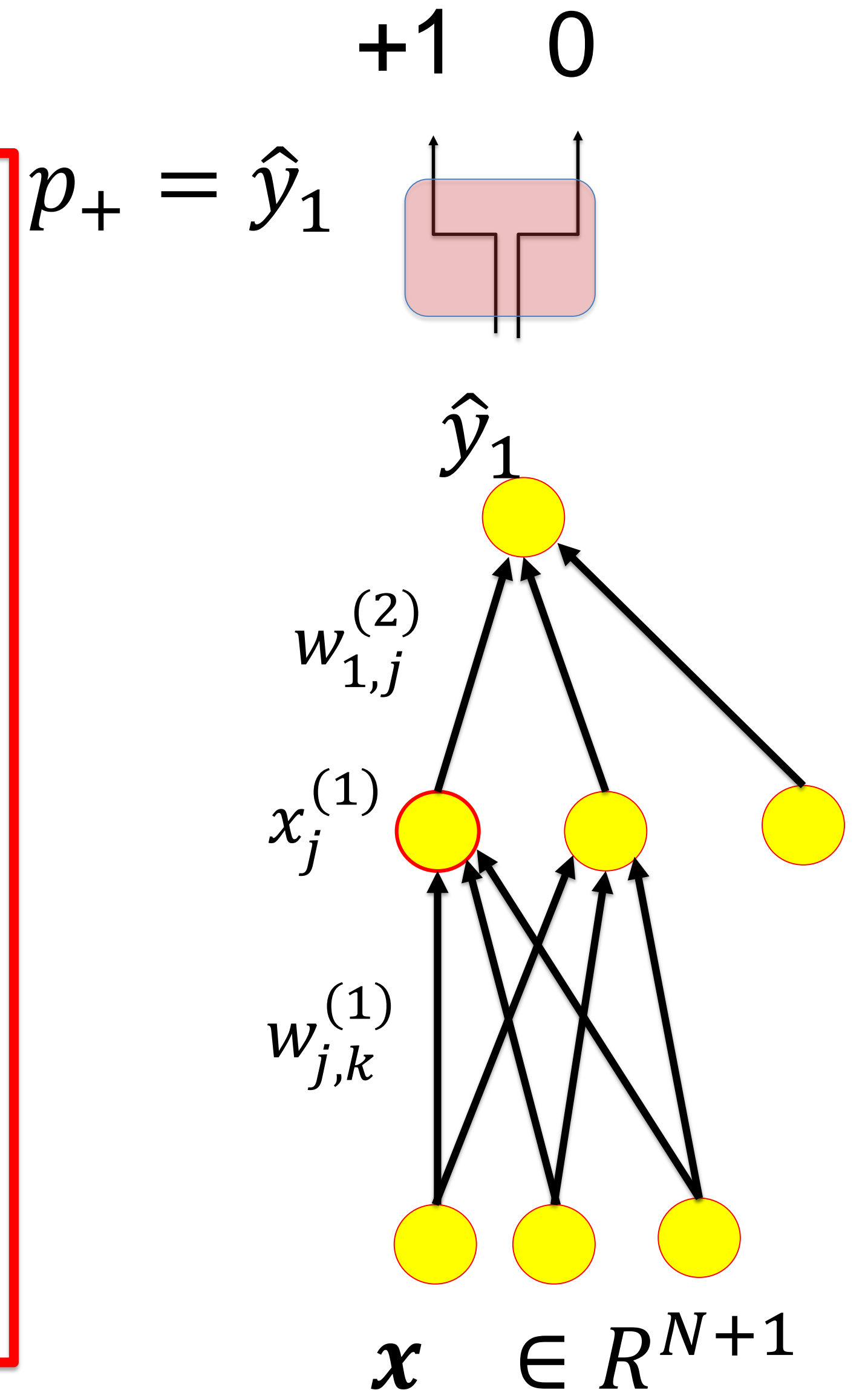
If we interpret the output

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

as a probability that the data \mathbf{x} belongs to class C_1 , then the drive/activation a of a sigmoidal output unit

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

can be interpreted as log-probability ratio.



Previous slide.

Summary of main finding.

Artificial Neural Networks

Statistical Classification by Deep Networks

Part 5: Multi-class problems

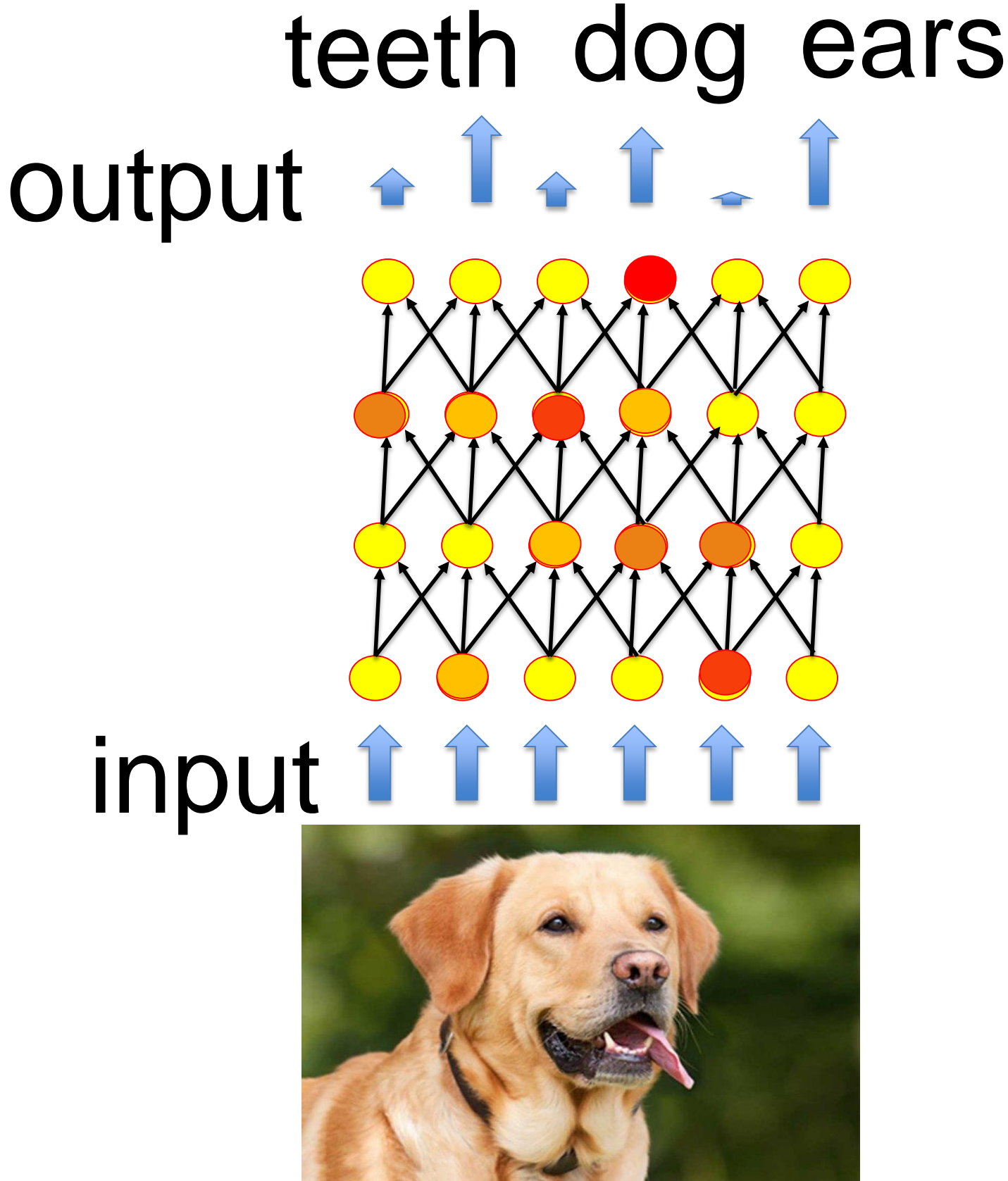
1. The statistical view: generative model
2. The likelihood of data under a model
3. Statistical interpretation of artificial neural networks
4. Sigmoidal as a natural output function
5. **Multi-class problems**

Previous slide.

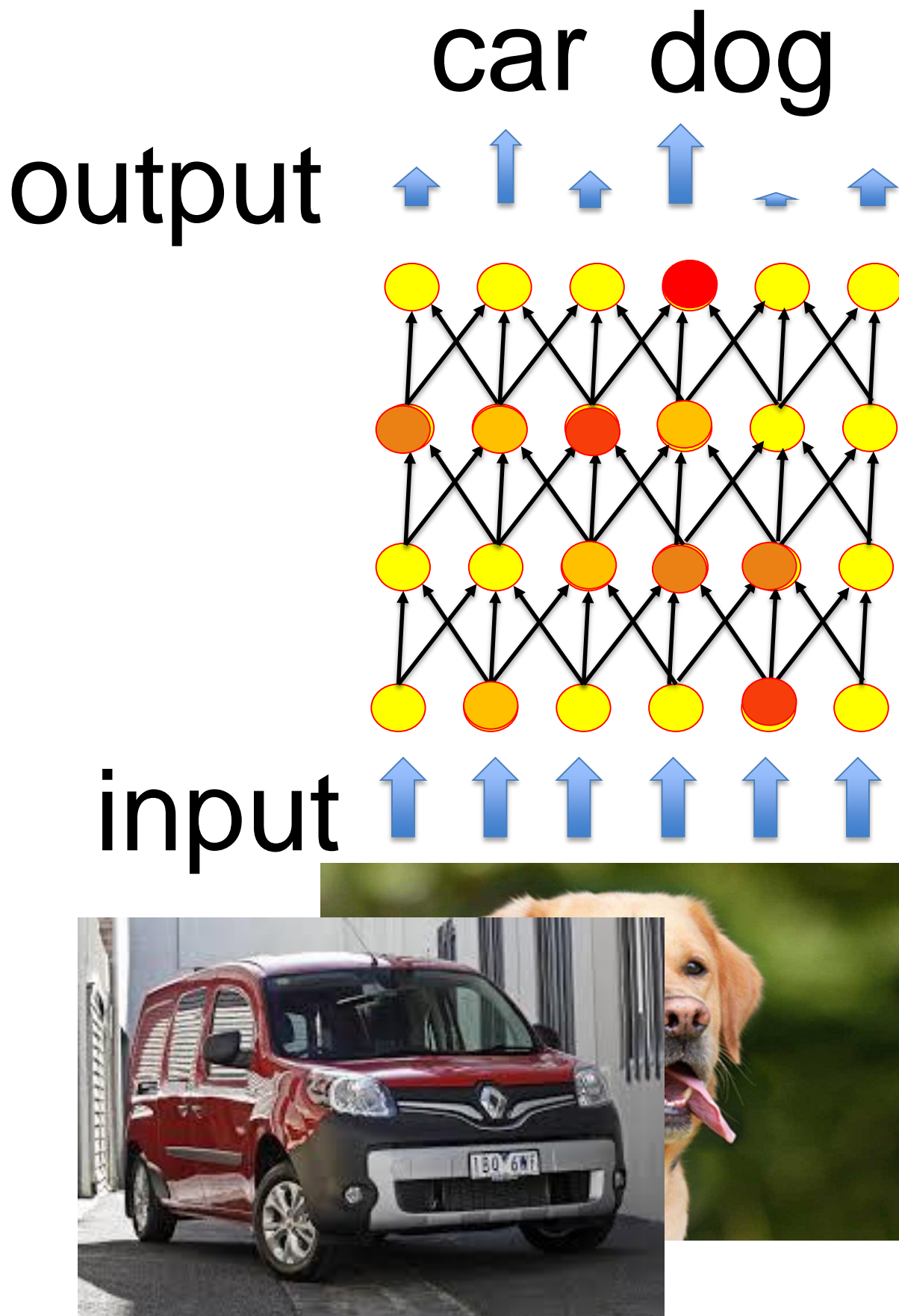
So far we worked with probabilities for a single class C, such as 'car' versus 'not car', represented by a neural network with a SINGLE output.

Multiple Classes

multiple attributes



mutually exclusive classes



Previous slide.

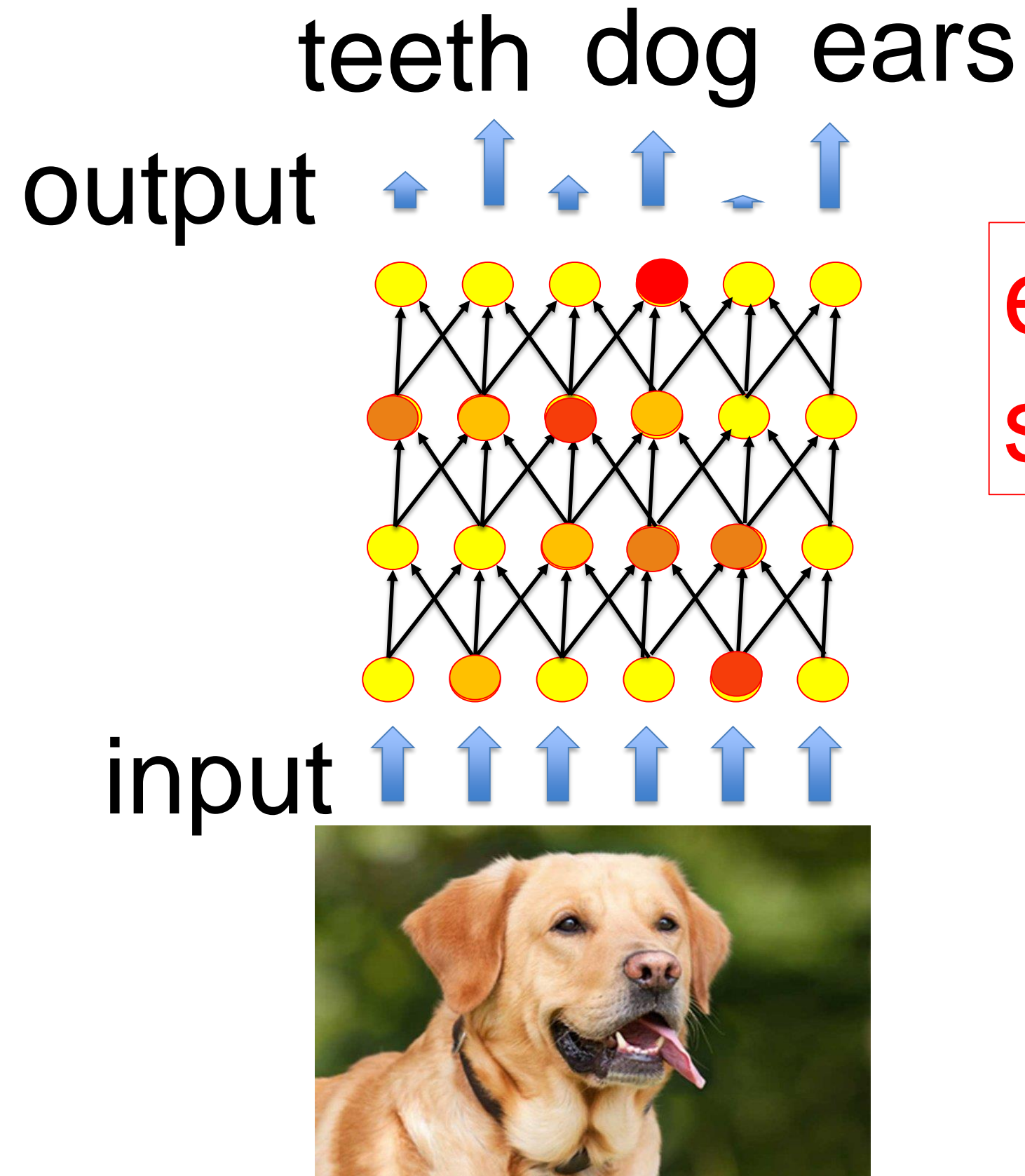
In the following we consider a network with multiple outputs. We need to distinguish between two different paradigms.

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.

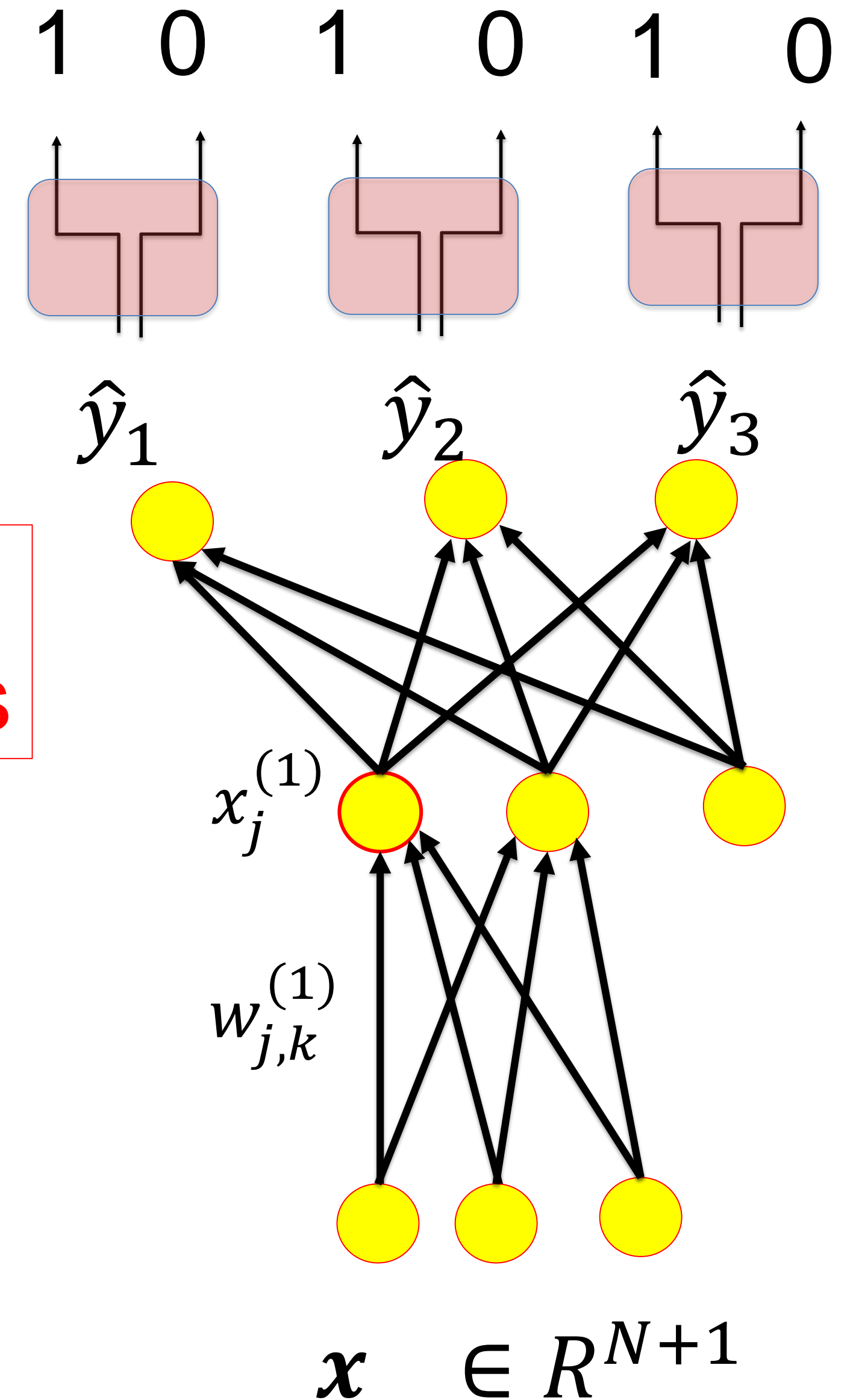
B (right): the outputs refer to mutually exclusive classes. For example, an image can be classified as either a car or a dog or a house, but not two at the same time. [This is an important assumption which implies that we exclude the case that the image could show one car and two dogs.]

Multiple Classes: Multiple attributes

Multiple attributes:



equivalent to several
single-class decisions



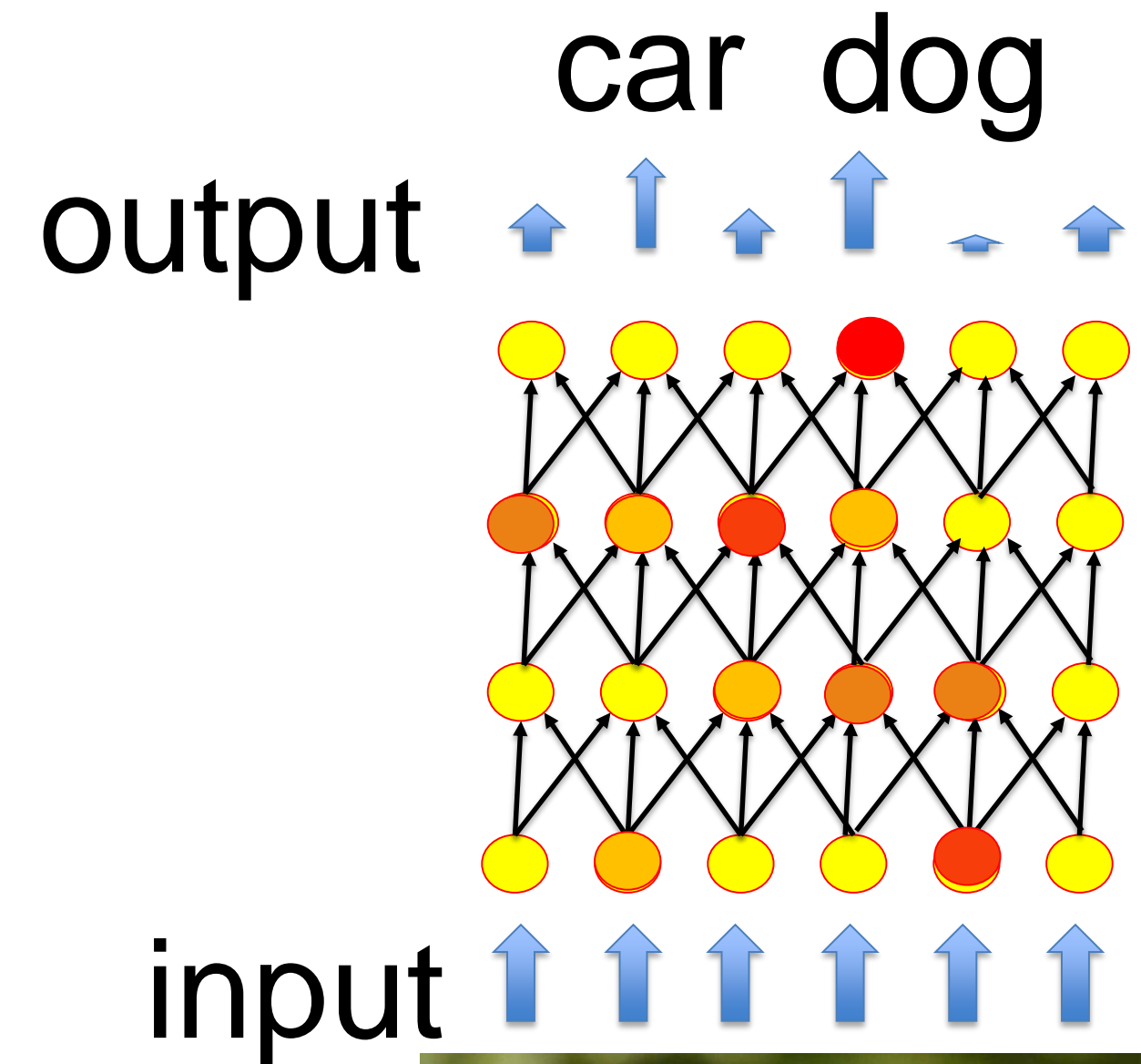
Previous slide.

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.

Multiple Classes: Mutually 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.

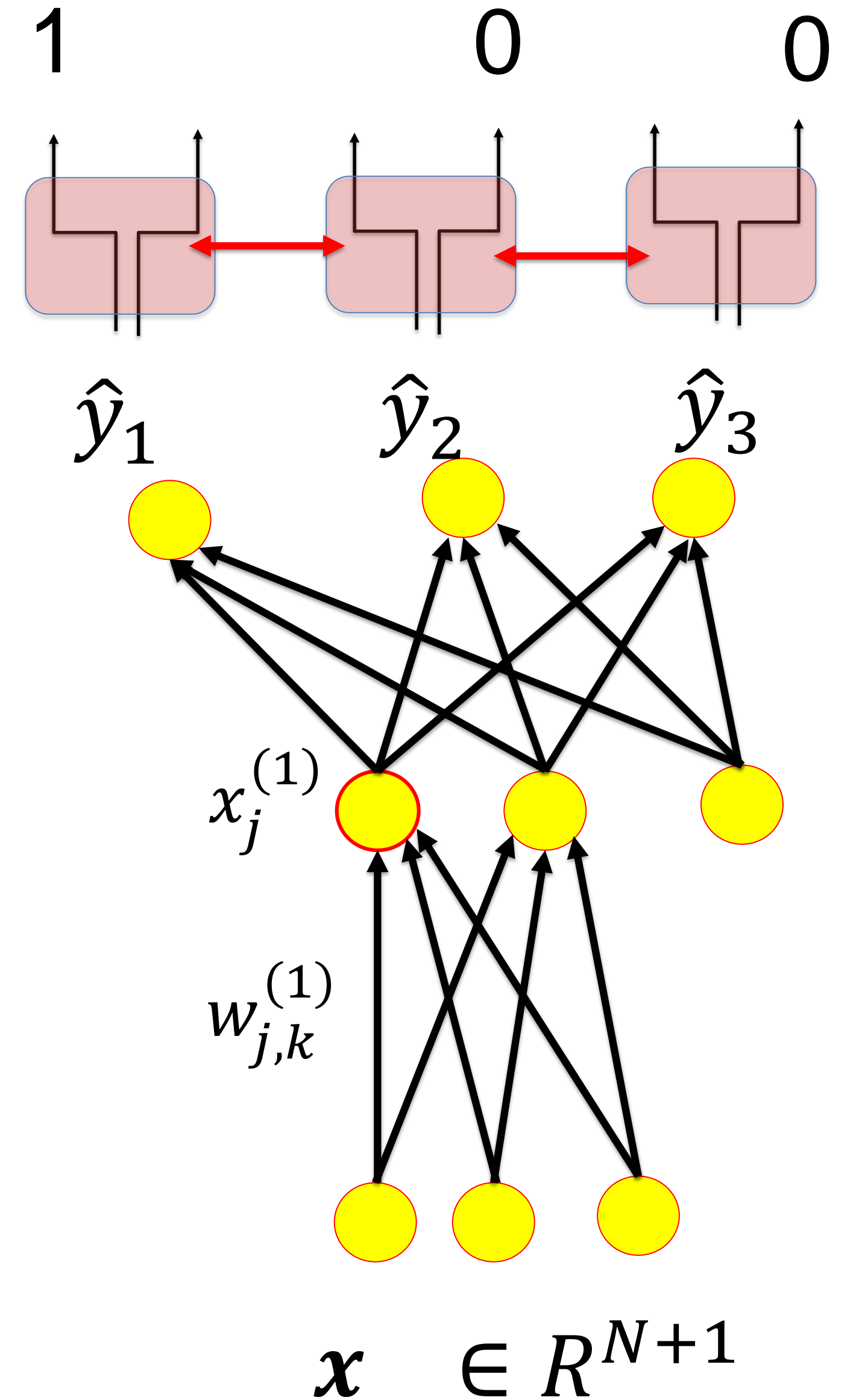
This implies that the outputs must interact. If one of the output shows 'true' the other must show 'not true'.

Exclusive Multiple Classes

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

1-hot-coding:

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



Previous slide.

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

Exclusive Multiple Classes

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

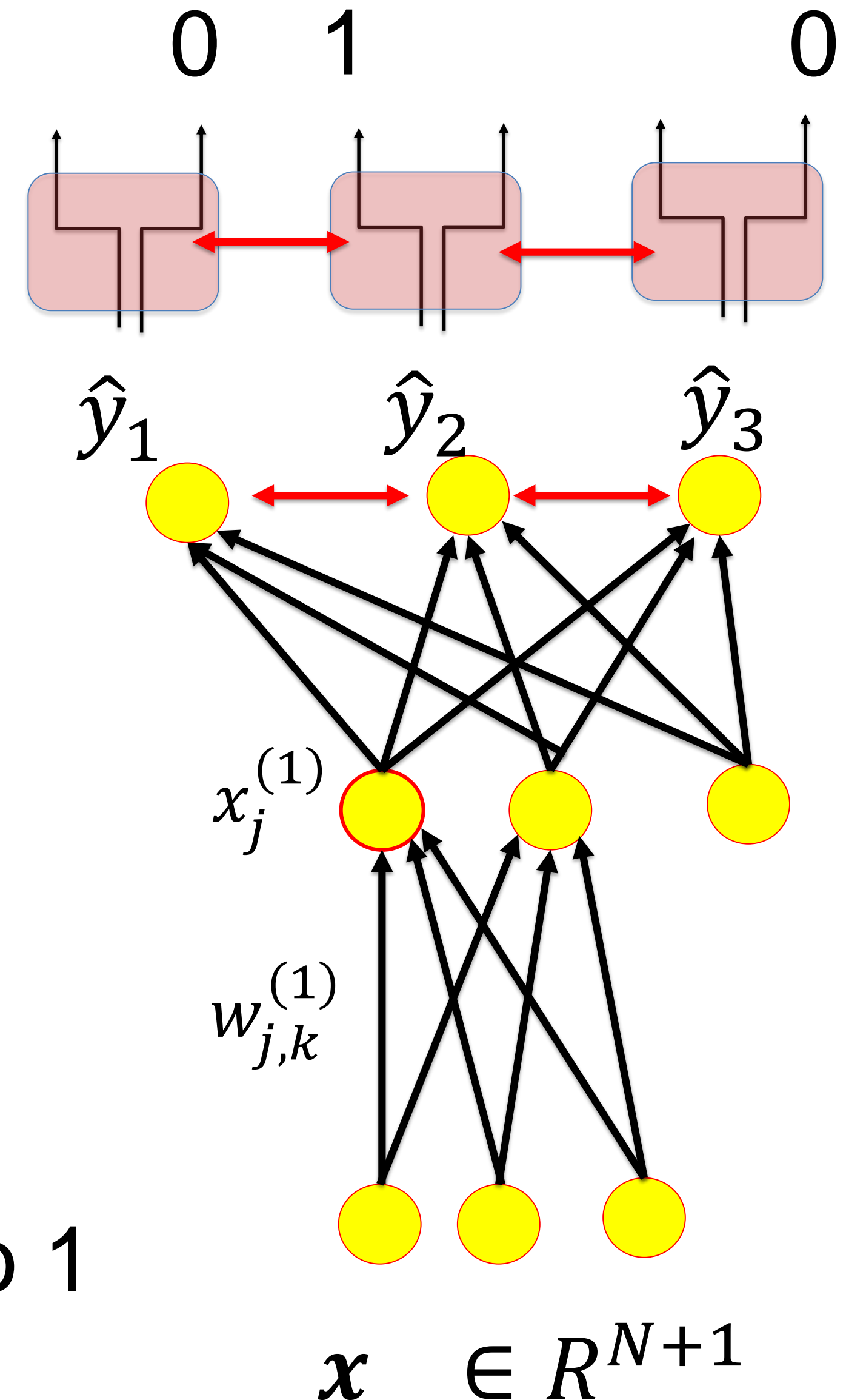
1-hot-coding:

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

Outputs are NOT independent:

$$\sum_{k=1}^K \hat{t}_k^\mu = 1 \quad \text{exactly one output is 1}$$

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



Previous slide.

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

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

In addition the PREDICTED output labels must also sum to one

$$\sum_{k=1}^K \hat{t}_k^{\mu} = 1$$

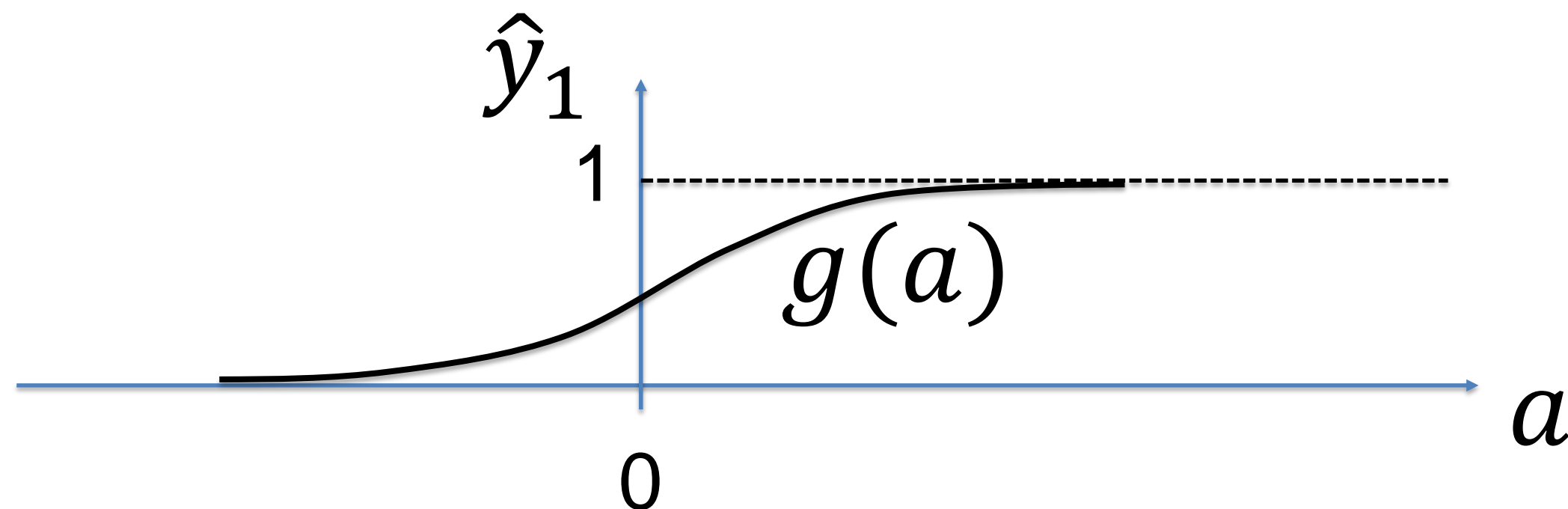
The question then is how we can implement both conditions.

Recall: Why sigmoidal output ?

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

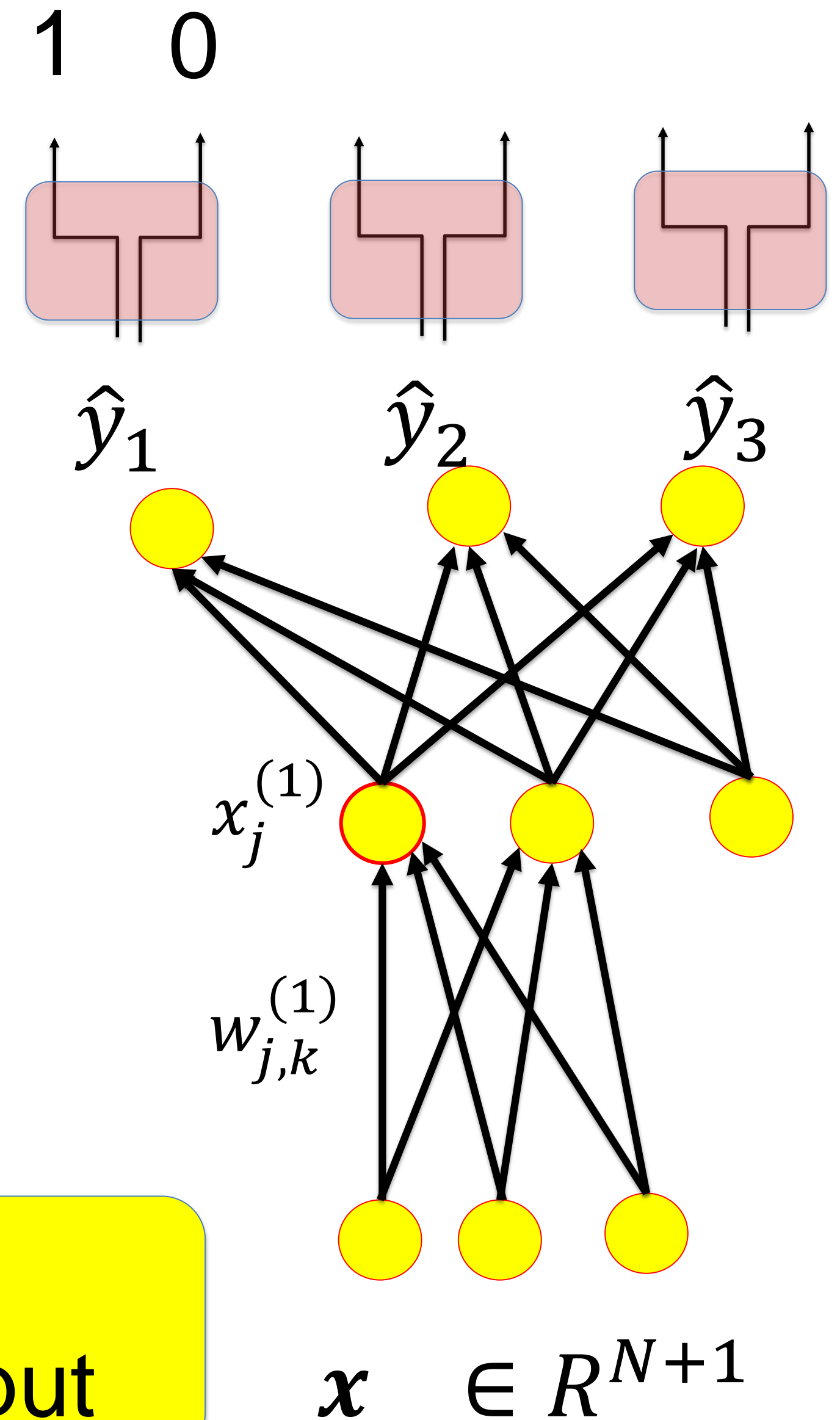
Observations (multiple-classes):

- Probabilities must sum to one!



AIM:

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,

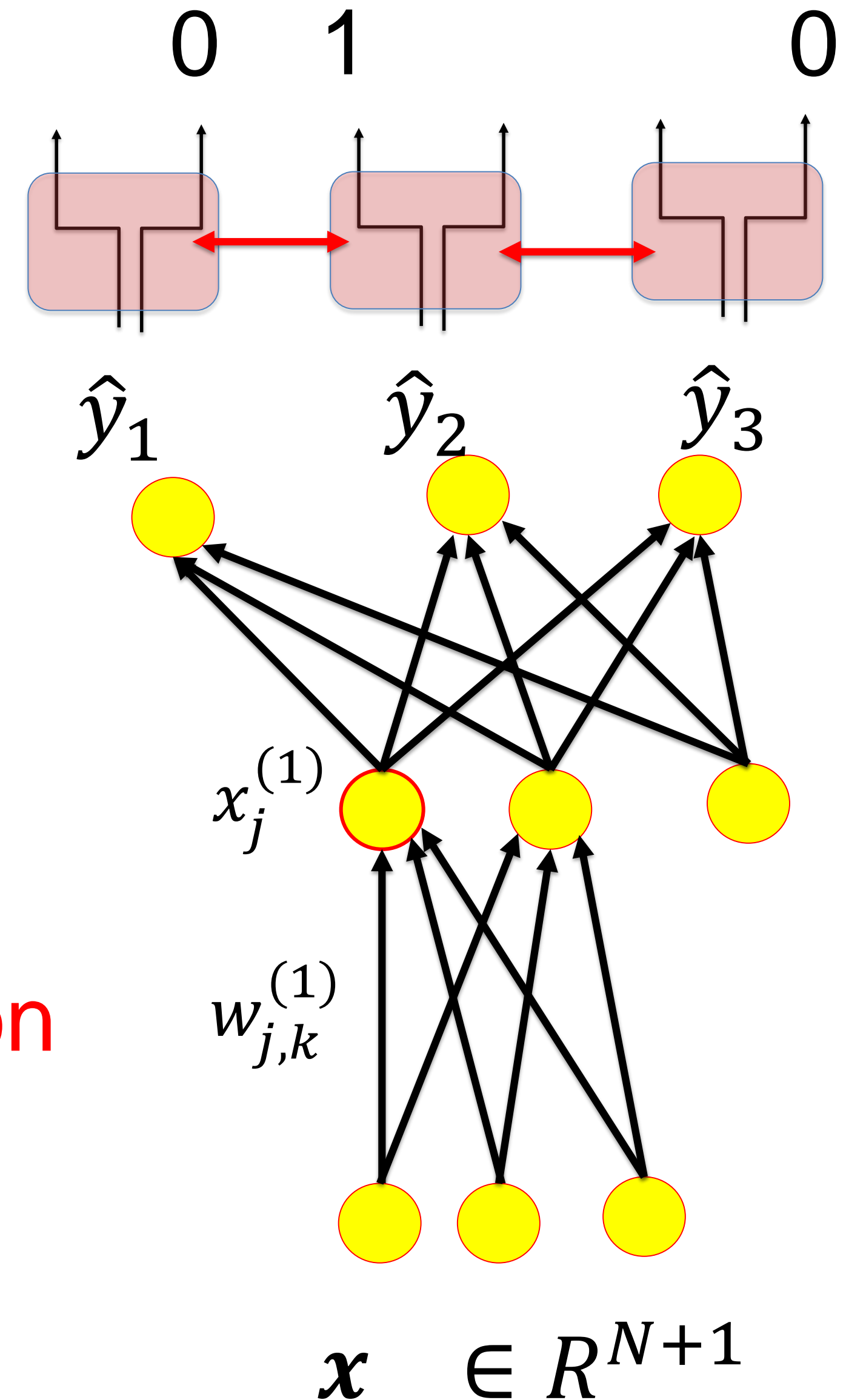
imply the 'softmax' output function.

Softmax output

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

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

‘softmax’:
Interaction of output units
that guarantees normalization



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

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

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

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

The right-hand side is called the 'softmax' function.

The denominator of the softmax function ensures normalization of probabilities to one.

Writing the numerator as an exponential ensures that probabilities are always positive.

Exclusive Multiple Classes

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

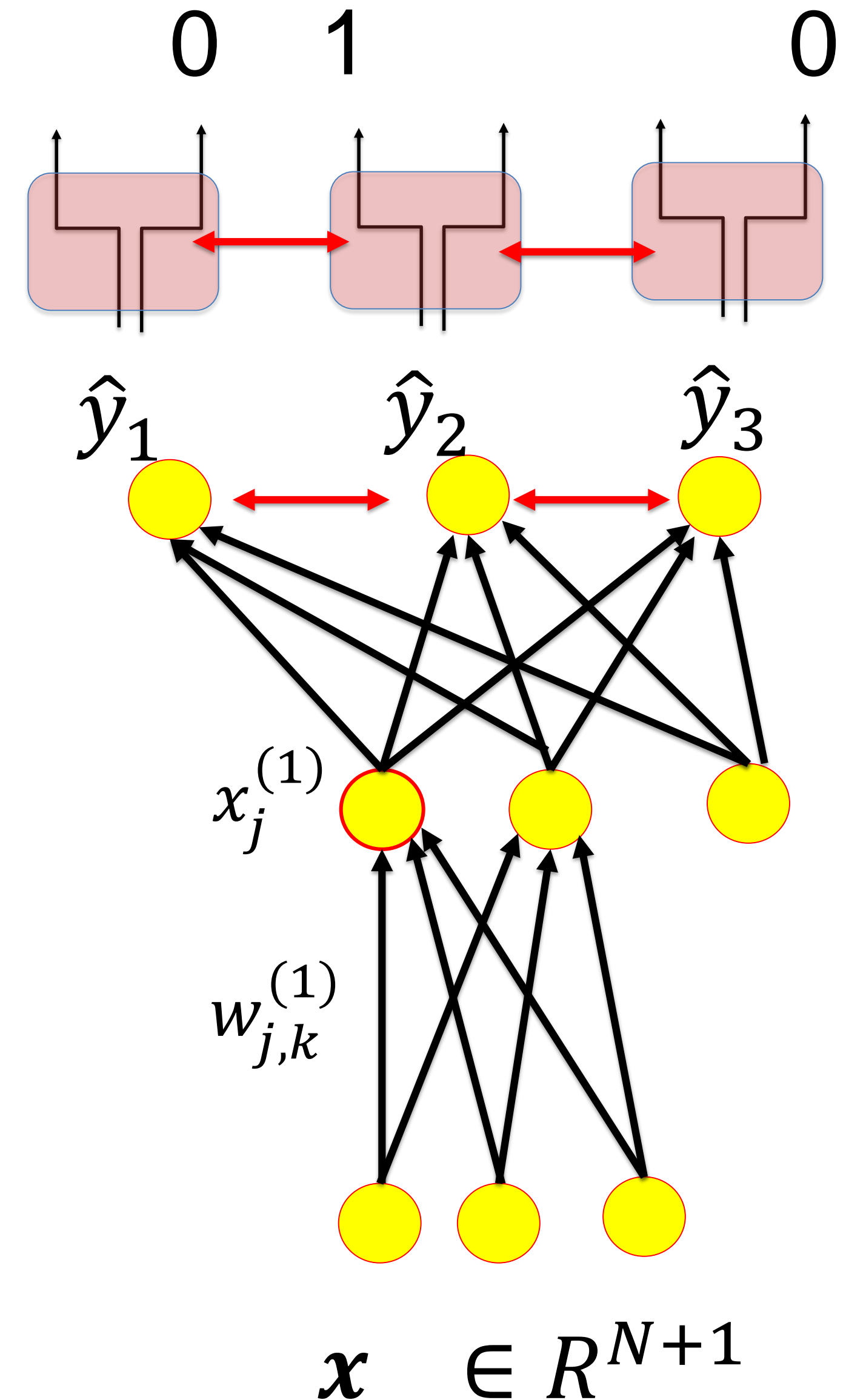
1-hot-coding:

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

Output labels are NOT independent:

$$\sum_{k=1}^K \hat{t}_k^\mu = 1 \quad \text{exactly one output is 1}$$

PREDICTED labels NOT independent



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.

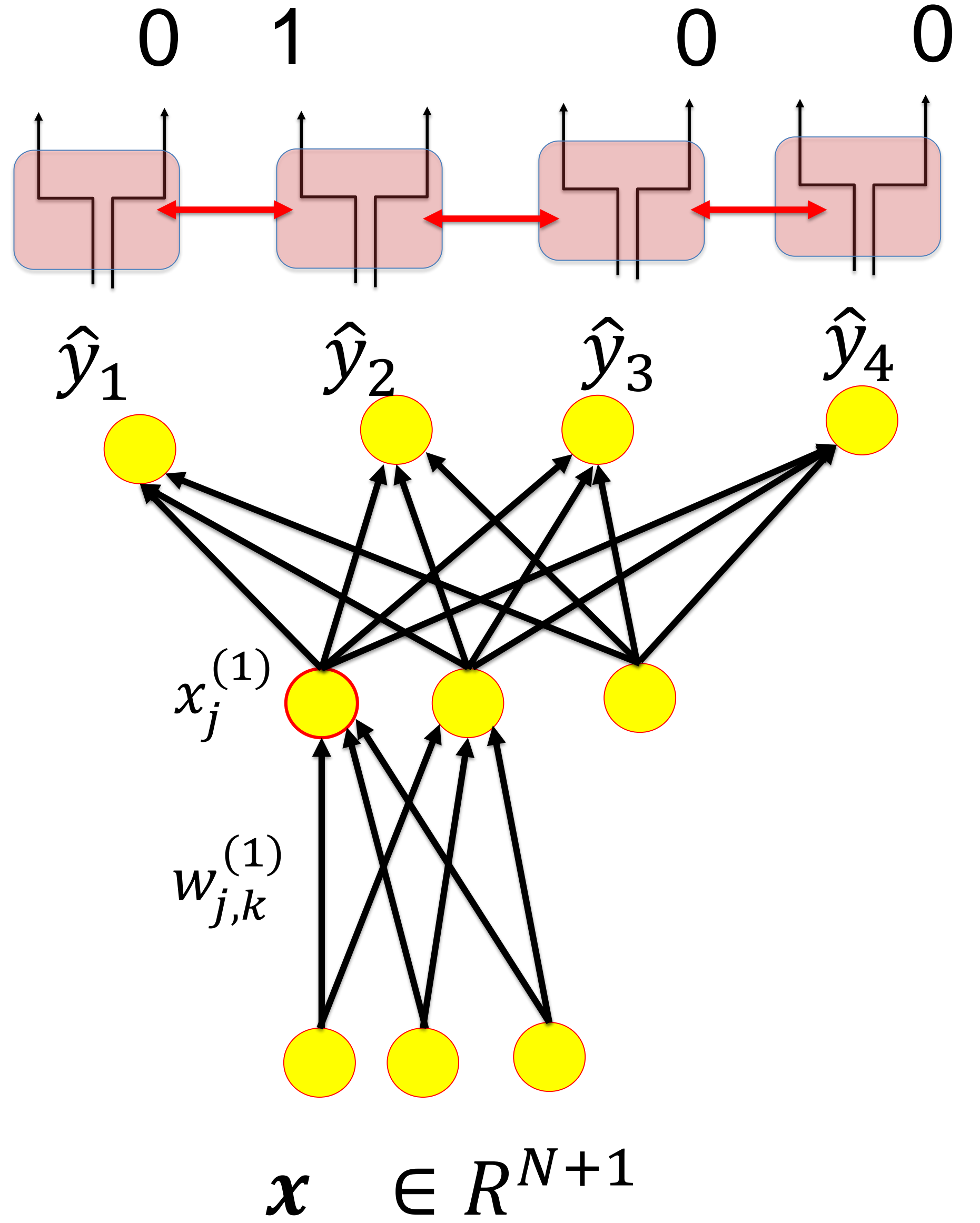
The result is also called cross-entropy error function.

output labels as symbols
mutually exclusive classes

4 symbols: A, B, C, D

prob	symbol
p_A	A= {1, 0, 0, 0}
p_B	B= {0, 1, 0, 0}
p_C	C= {0, 0, 1, 0}
p_D	D= {0, 0, 0, 1}
arbitrary	$\mathbf{t} = \{t_1, t_2, t_3, t_4\}$

$$p_A + p_B + p_C + p_D = 1$$



Previous slide:

1-hot coding implies that the symbol for each target is a string such as

$\{0,0, \dots, 0, 1, 0, \dots 0\}$.

In the case of four mutually exclusive classes, class C is represented by the string

$\{t_1, \dots, t_4\} = \{0, 0, 1, 0\}$.

We can write the probability p_C as

$$p_C = p_A^0 \cdot p_B^0 \cdot p_C^1 \cdot p_D^0$$

and use this to derive the log-likelihood function.

output labels as symbols
mutually exclusive classes

4 symbols: A, B, C, D

prob	symbol
p_A	A = {1, 0, 0, 0}
p_B	B = {0, 1, 0, 0}
p_C	C = {0, 0, 1, 0}
p_D	D = {0, 0, 0, 1}
arbitrary	$\mathbf{t} = \{t_1, t_2, t_3, t_4\}$

$$p_A + p_B + p_C + p_D = 1$$

Your notes.

Summary: Cross-entropy error for Multiclass problems

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

Minimize* the **cross-entropy**

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

parameters= all weights, all layers

*Minimization under the constraint:

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

→ softmax

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

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

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

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

Wulfram Gerstner

EPFL, Lausanne, Switzerland

Statistical Classification by Deep Networks

Part 6: Summary and Quiz

1. The statistical view: generative model
2. The likelihood of data under a model
3. Statistical interpretation of artificial neural networks
4. Sigmoidal as a natural output function
5. Multi-class problems
6. **Summary and Quiz**

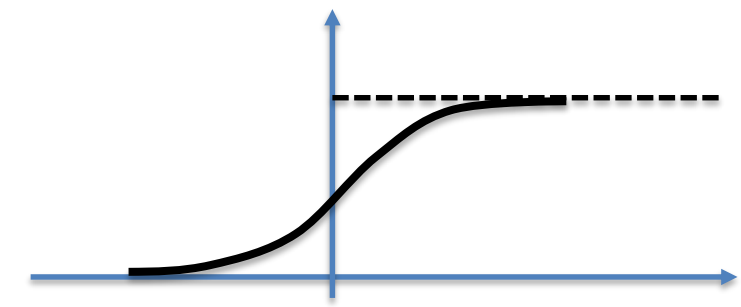
Previous slide.

Summary of the probabilistic interpretation.

Modern Neural Networks for classification

output layer

use sigmoidal unit (single-class)



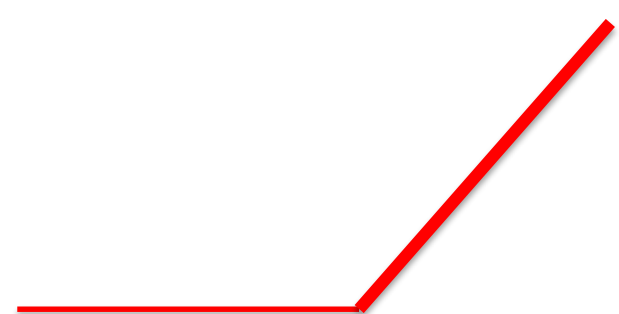
$$\hat{y}_1 = g(a) = \frac{1}{1 + e^{-a}}$$

or softmax (exclusive multi-class)

$$\hat{y}_k = g(a) = \frac{e^{a_k}}{\sum_i e^{a_i}}$$

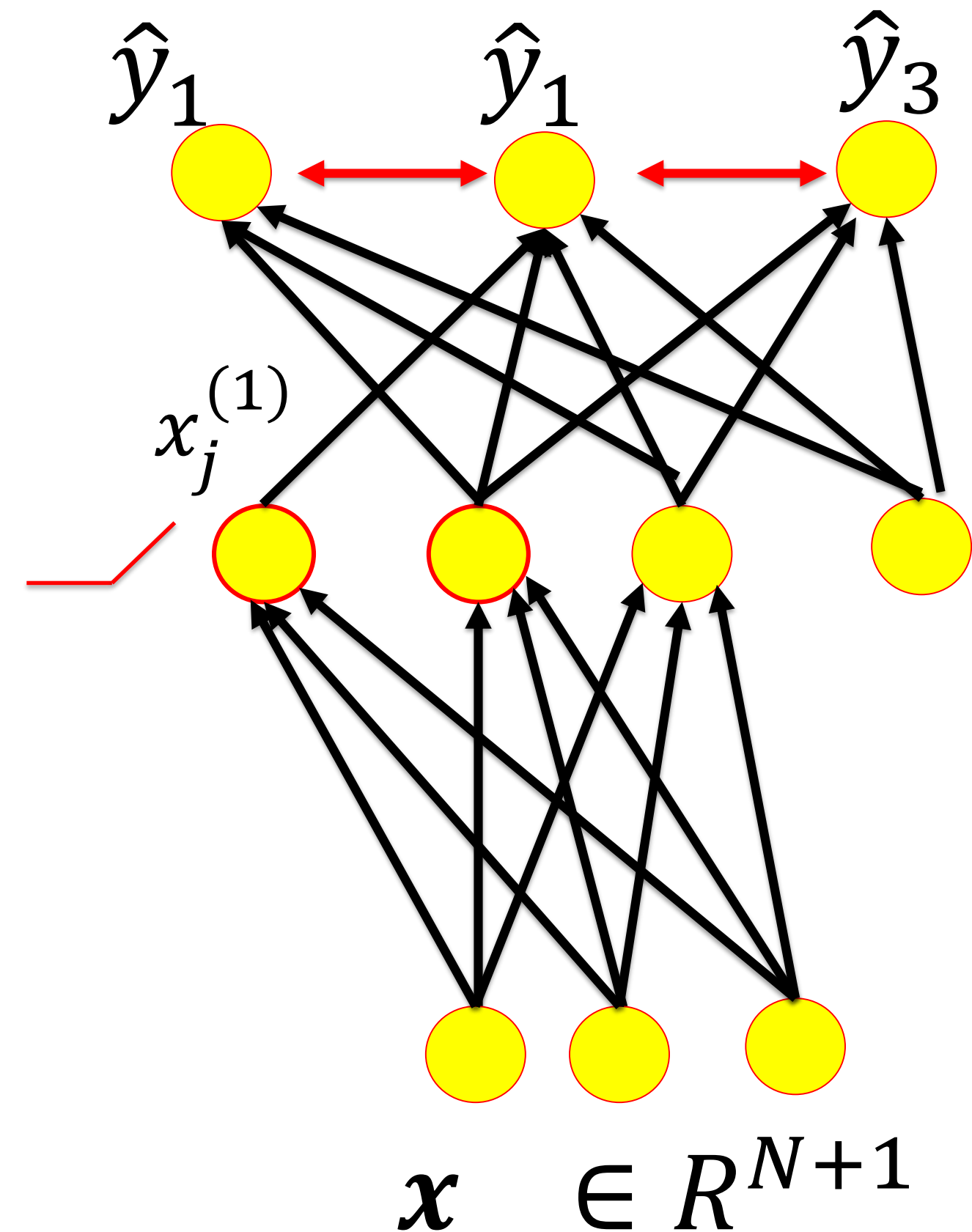
hidden layers

use rectified linear unit



$$f(x) = x \text{ for } x > 0$$

$$f(x) = 0 \text{ for } x \leq 0$$



Previous slide.

The statistical interpretation of Neural Networks suggest to use for classification tasks a sigmoidal unit and for exclusive classes a softmax function.

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

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.

Modern Neural Networks for classification

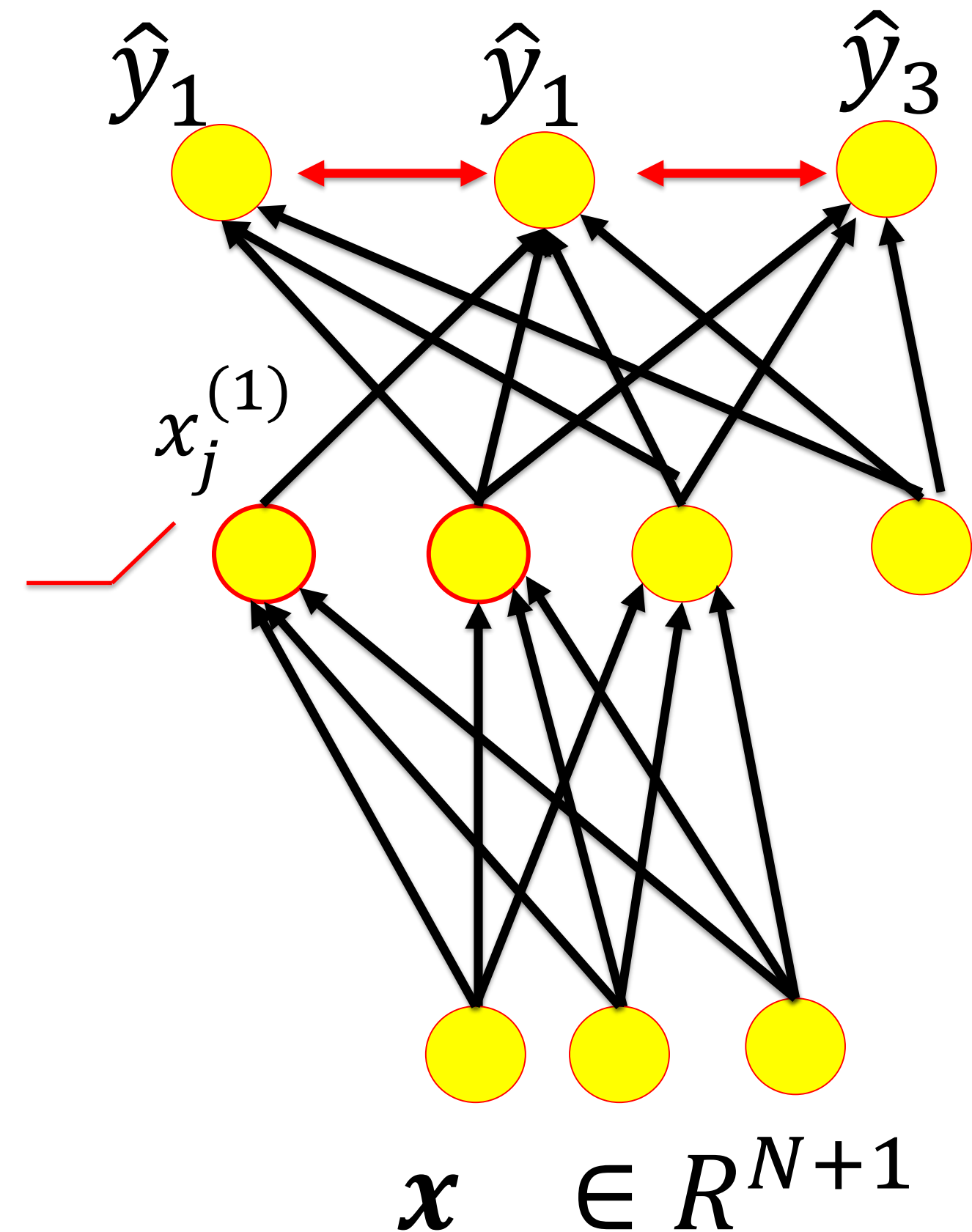
Best error function in context
of classification:

cross-entropy loss (single-class)

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

Cross-entropy loss:
mutually exclusive classes

$$E(\mathbf{w}) = -\sum_k \sum_{\mu} [t_k^{\mu} \ln \hat{y}_k^{\mu}]$$



QUIZ: Modern Neural Networks in a statistical setting

- softmax unit should be used for exclusive multi-class in an output layer in problems with 1-hot coding
- sigmoidal units should be used for multiple attribute-classification problems
- two-class problems (mutually exclusive) are the same as single-class problems
- multiple-attribute-class problems are treated as multiple-single-class problems
- the above choices guarantee that the outputs can be interpreted as probabilities
- we need to be careful, if we want to interpret the output as a probability

QUIZ: Modern Neural Networks in a statistical setting

[] Suppose we minimize, in a multi-class setting the cross-entropy loss with respect to the model parameters (weights and thresholds of a neural network). THEN the final set of parameters at the global minimum corresponds to the maximum-likelihood solution in the statistical sense.

We have a problem **with two mutually exclusive classes**.

Albert uses **soft-max** for the two outputs of his neural network and minimizes the cross-entropy loss

$$E(\mathbf{w}) = - \sum_k \sum_{\mu} [t_k^{\mu} \ln \hat{y}_k^{\mu}]$$

Berta works with a single output, uses a **sigmoidal** unit for it, and minimizes the loss

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

[] At the global minimum, the two solutions are completely equivalent.

[] Both approaches work, but the solution of Berta is (slightly) better.

[] Both approaches work, but the solution of Albert is (slightly) better.

Your notes..

Artificial Neural Networks

Wulfram Gerstner

EPFL, Lausanne, Switzerland

Statistical classification by deep networks

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 (restrictive) conditions we may interpret the output as a probability

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