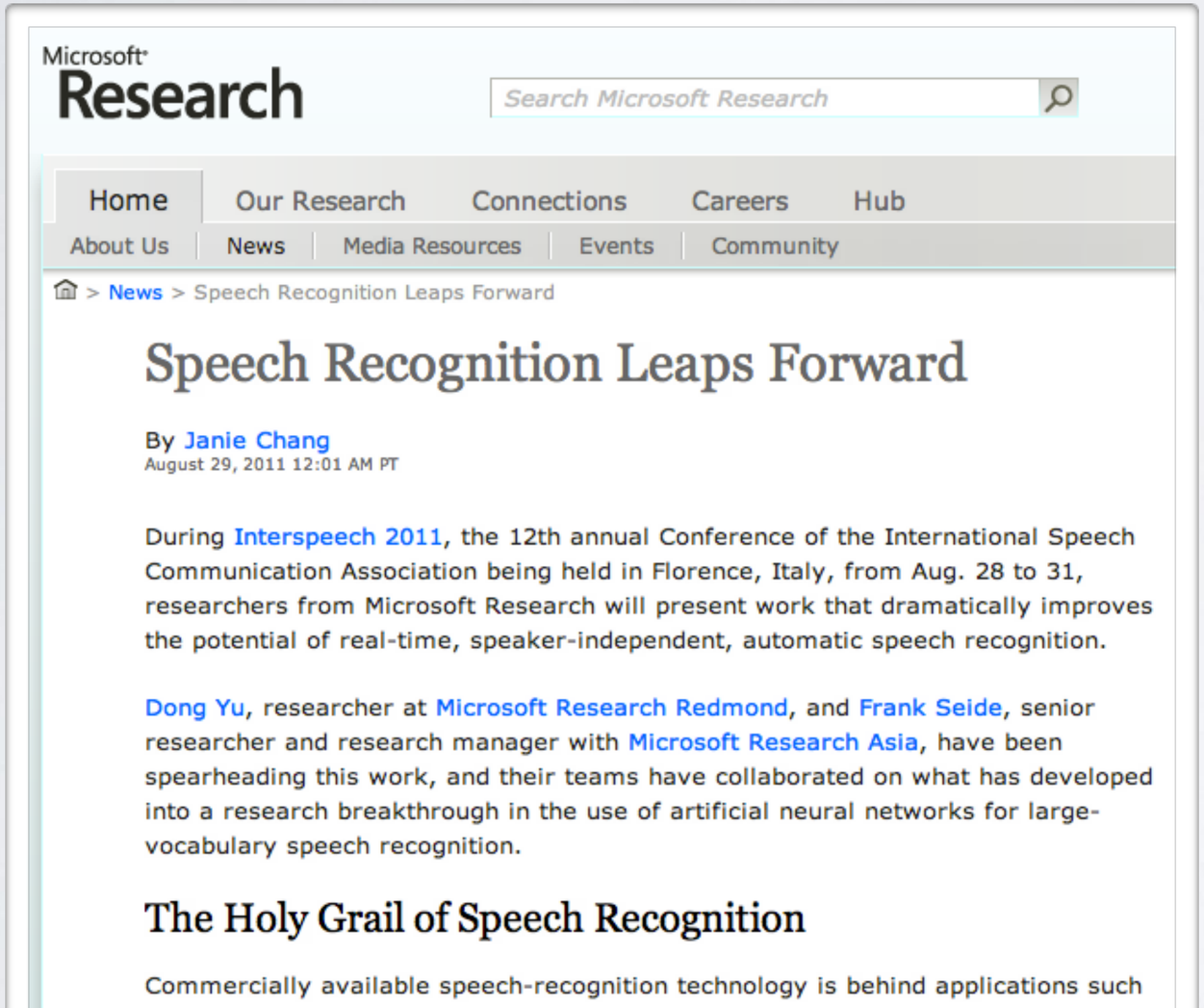
Neural Networks

Hugo Larochelle (@hugo_larochelle)
Google Brain

DEEP LEARNING

Topics: success story: speech recognition



Microsoft Research

Search Microsoft Research

Home | Our Research | Connections | Careers | Hub

About Us | News | Media Resources | Events | Community

> News > Speech Recognition Leaps Forward

Speech Recognition Leaps Forward

By Janie Chang
August 29, 2011 12:01 AM PT

During Interspeech 2011, the 12th annual Conference of the International Speech Communication Association being held in Florence, Italy, from Aug. 28 to 31, researchers from Microsoft Research will present work that dramatically improves the potential of real-time, speaker-independent, automatic speech recognition.

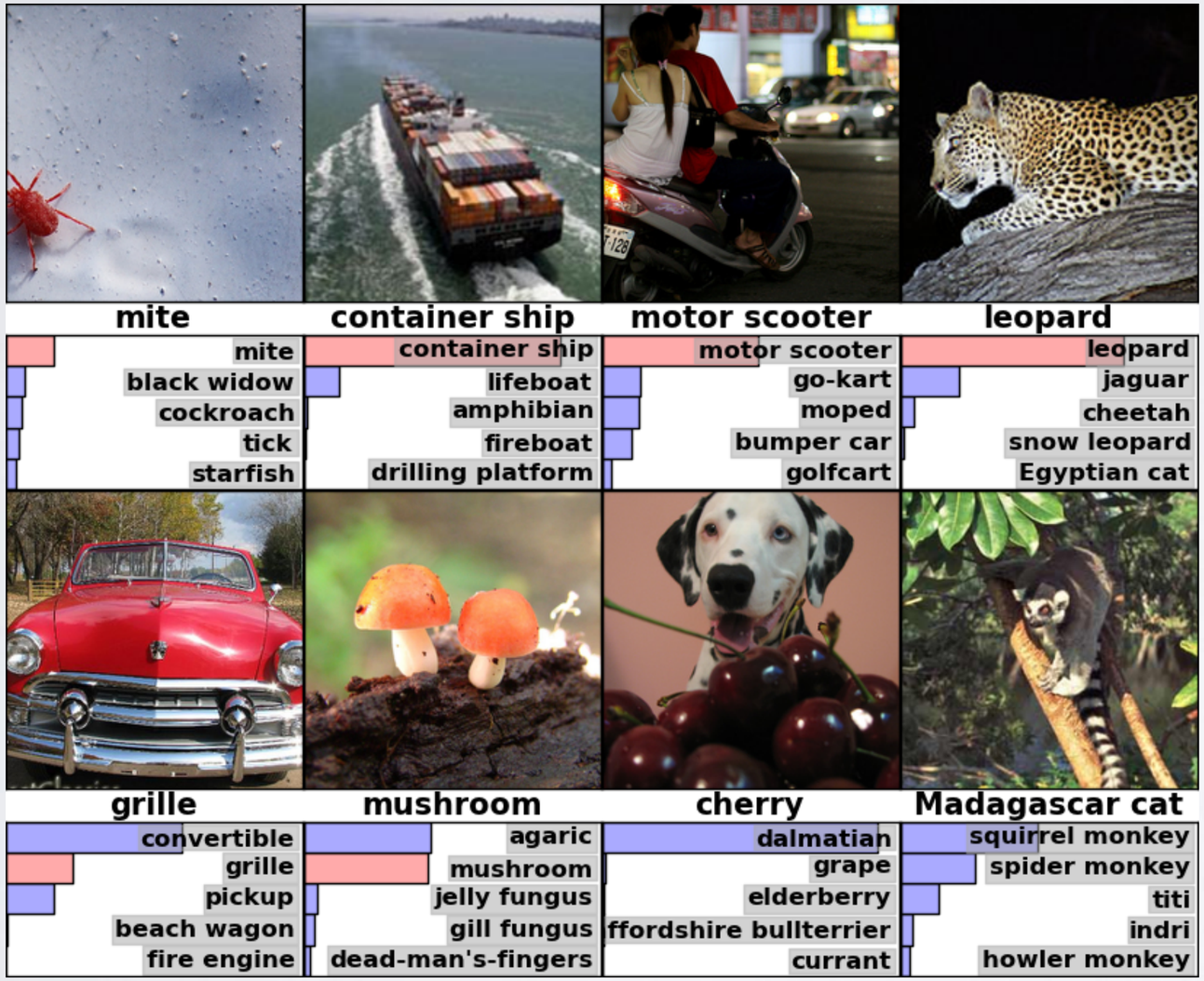
Dong Yu, researcher at Microsoft Research Redmond, and Frank Seide, senior researcher and research manager with Microsoft Research Asia, have been spearheading this work, and their teams have collaborated on what has developed into a research breakthrough in the use of artificial neural networks for large-vocabulary speech recognition.

The Holy Grail of Speech Recognition

Commercially available speech-recognition technology is behind applications such

DEEP LEARNING

Topics: success story: computer vision



ARTIFICIAL NEURON

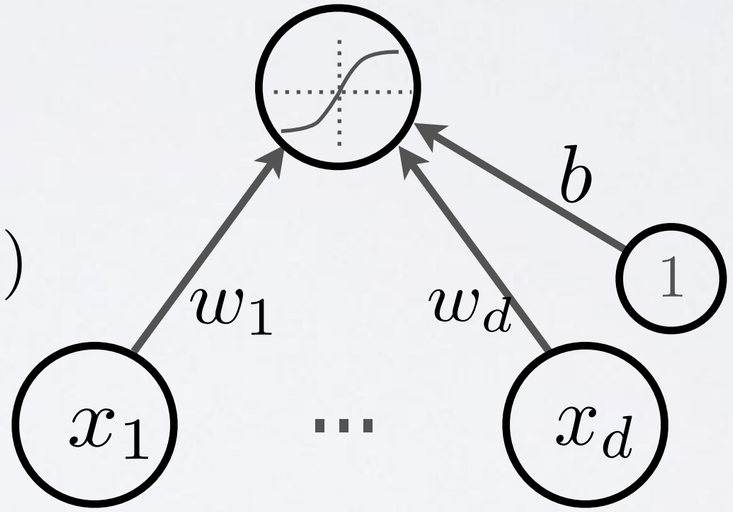
Topics: connection weights, bias, activation function

- Neuron pre-activation (or input activation):

$$a(\mathbf{x}) = b + \sum_i w_i x_i = b + \mathbf{w}^\top \mathbf{x}$$

- Neuron (output) activation

$$h(\mathbf{x}) = g(a(\mathbf{x})) = g(b + \sum_i w_i x_i)$$



$\mathbf{w}$ are the connection weights

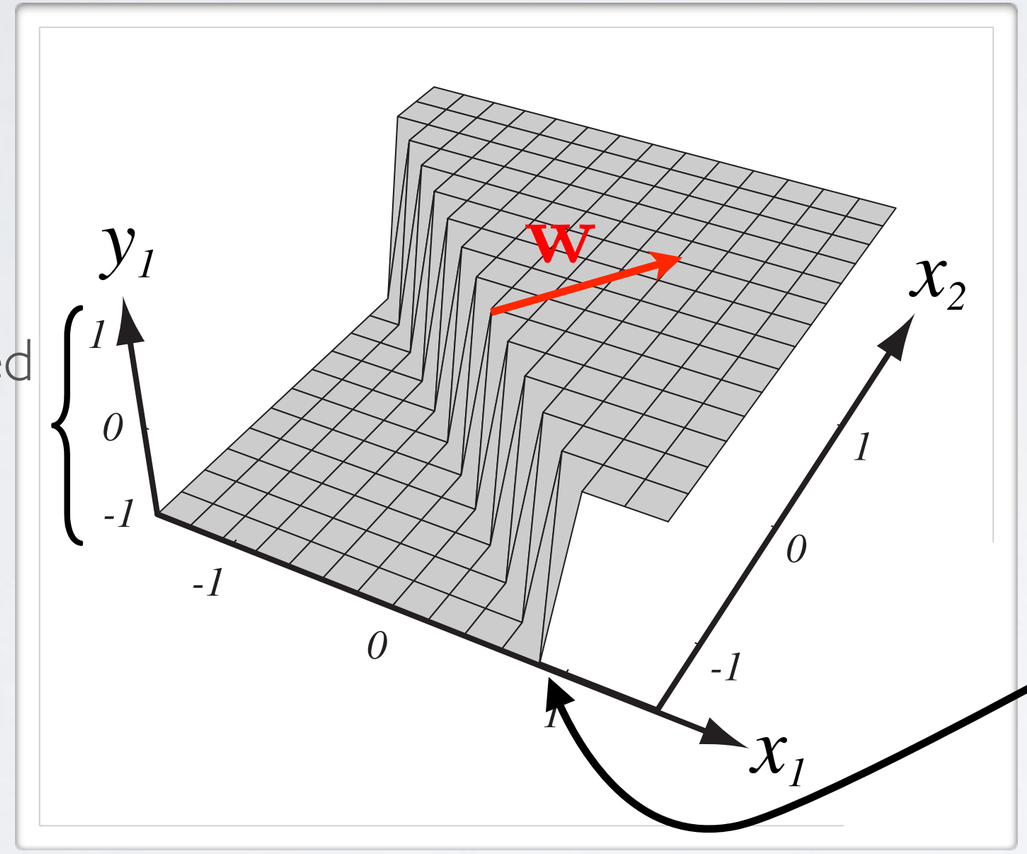
b is the neuron bias

$g(\cdot)$ is called the activation function

ARTIFICIAL NEURON

Topics: connection weights, bias, activation function

range determined by $g(\cdot)$

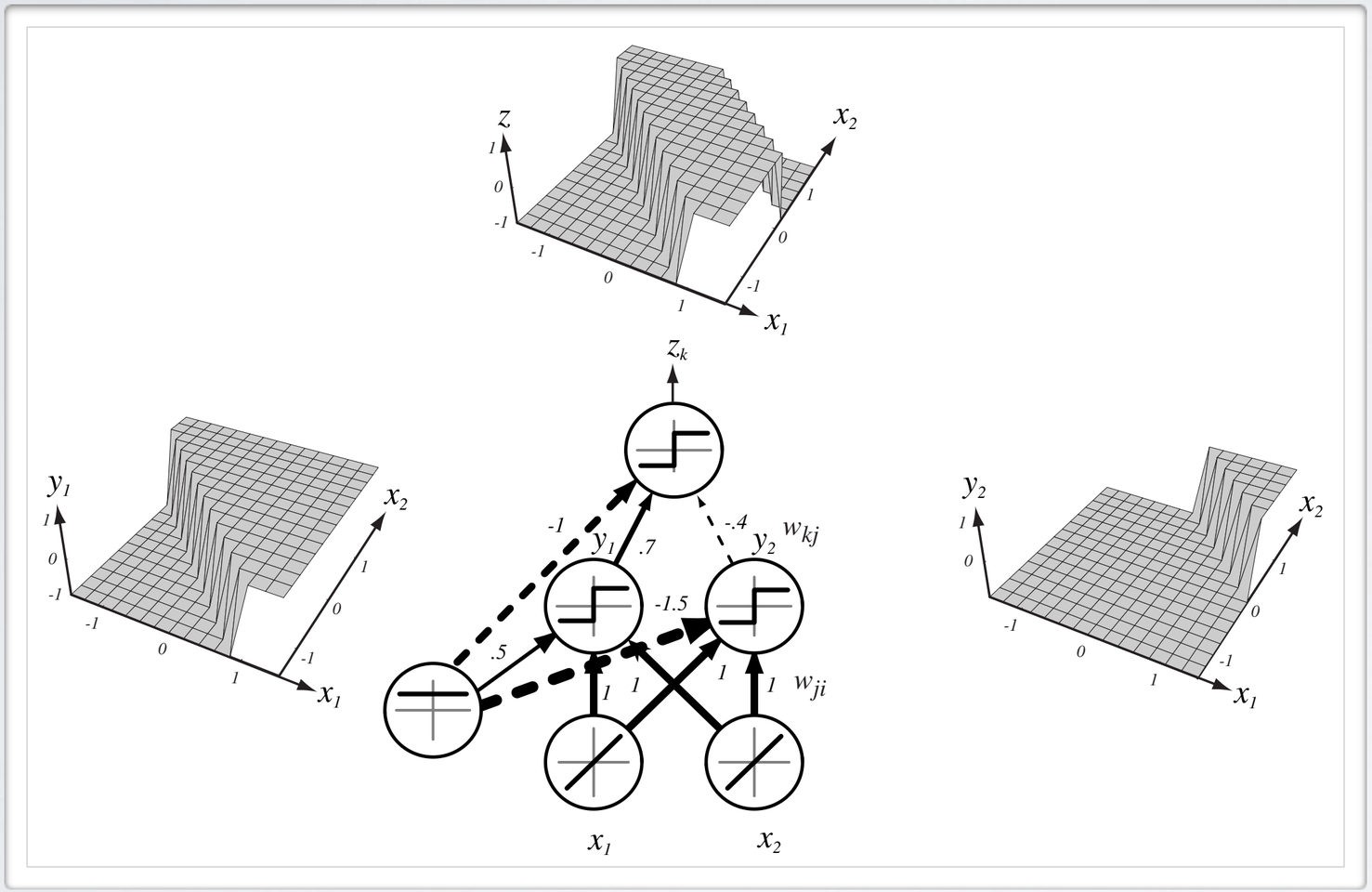


bias b only changes the position of the riff

(from Pascal Vincent's slides)

CAPACITY OF NEURAL NETWORK

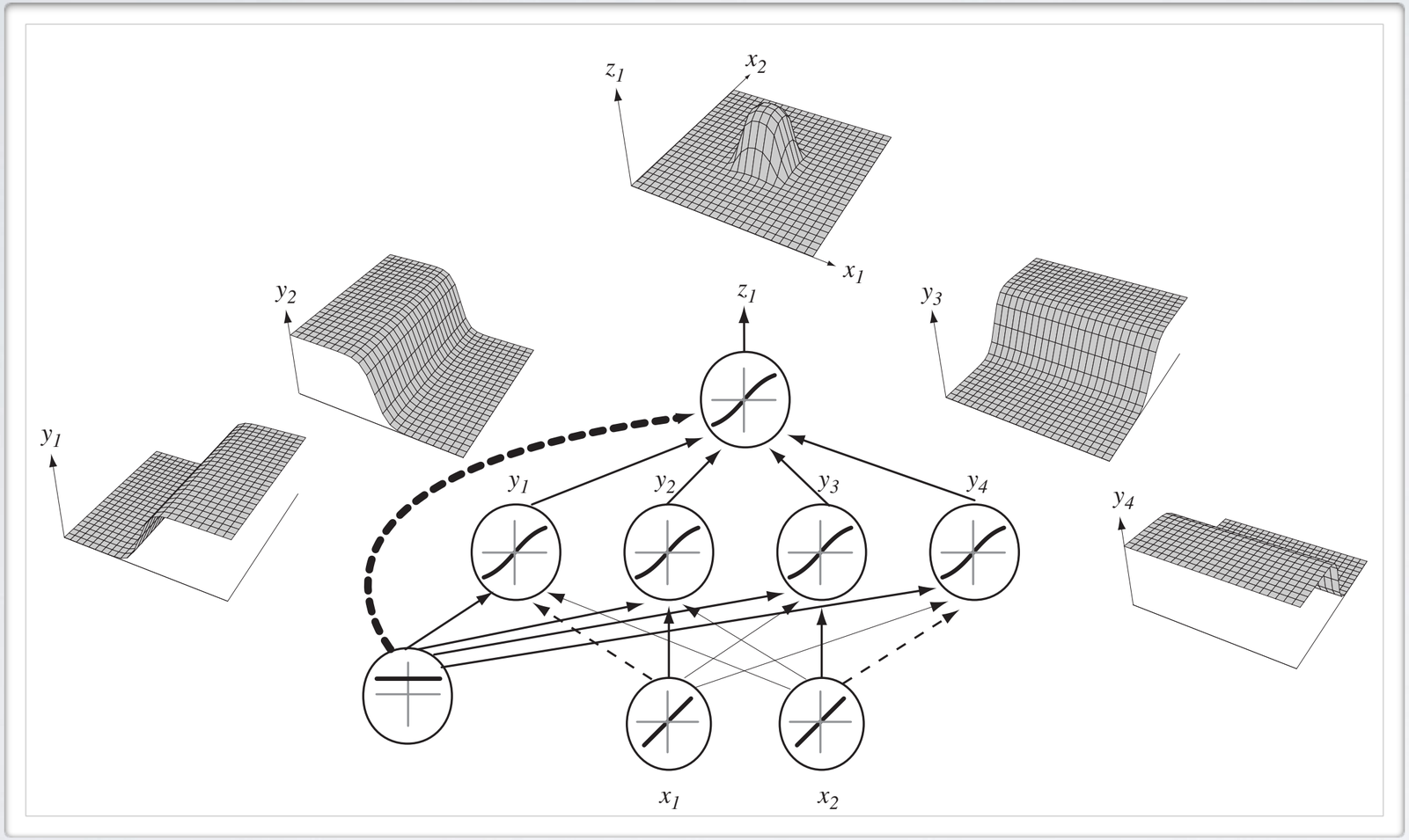
Topics: single hidden layer neural network



(from Pascal Vincent's slides)

CAPACITY OF NEURAL NETWORK

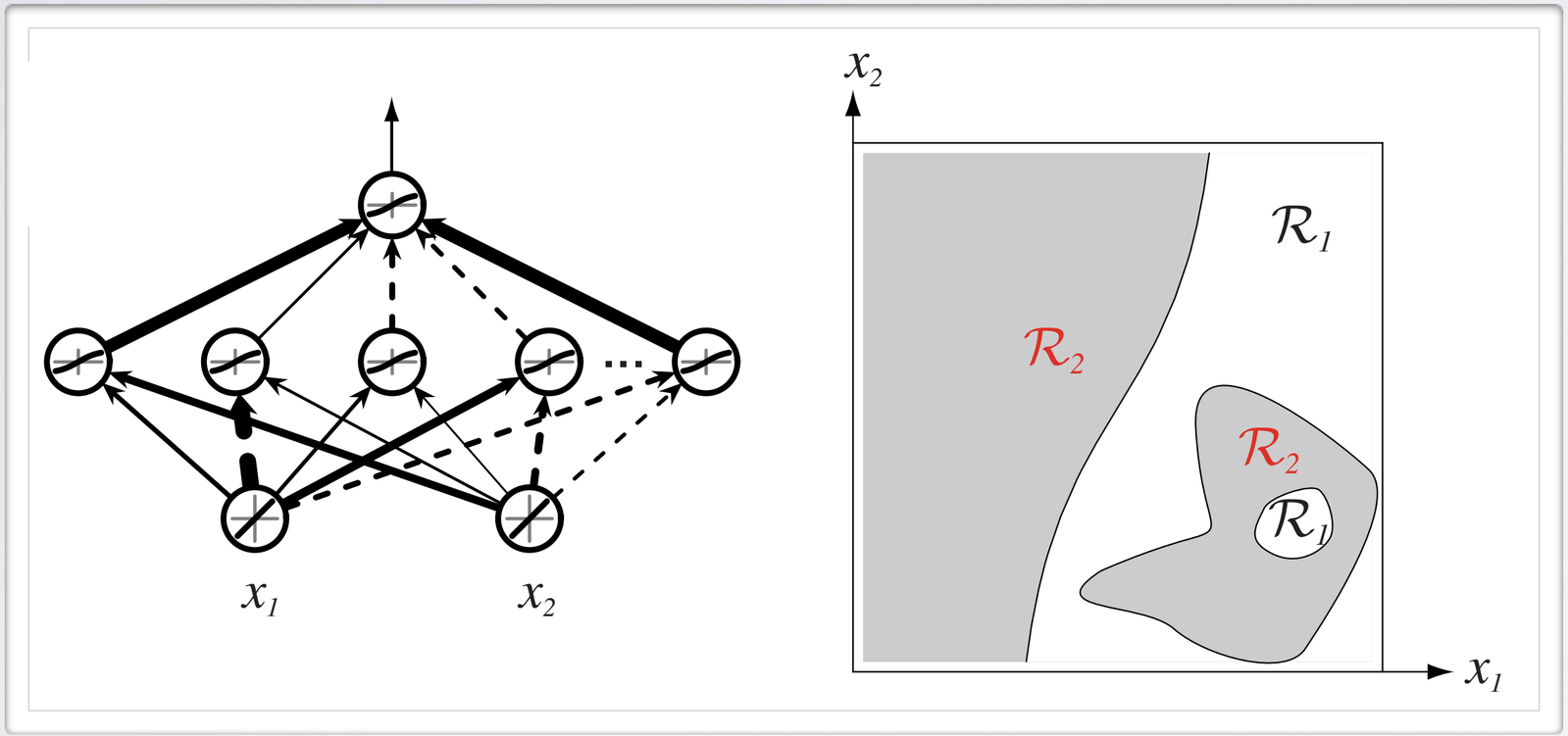
Topics: single hidden layer neural network



(from Pascal Vincent's slides)

CAPACITY OF NEURAL NETWORK

Topics: single hidden layer neural network



(from Pascal Vincent's slides)

CAPACITY OF NEURAL NETWORK

Topics: universal approximation

- Universal approximation theorem (Hornik, 1991):
  - "a single hidden layer neural network with a linear output unit can approximate any continuous function arbitrarily well, given enough hidden units"
- The result applies for sigmoid, tanh and many other hidden layer activation functions
- This is a good result, but it doesn't mean there is a learning algorithm that can find the necessary parameter values!

NEURAL NETWORK

Topics: multilayer neural network

- Could have L hidden layers:
  - layer pre-activation for $k>0$ $(\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x})$

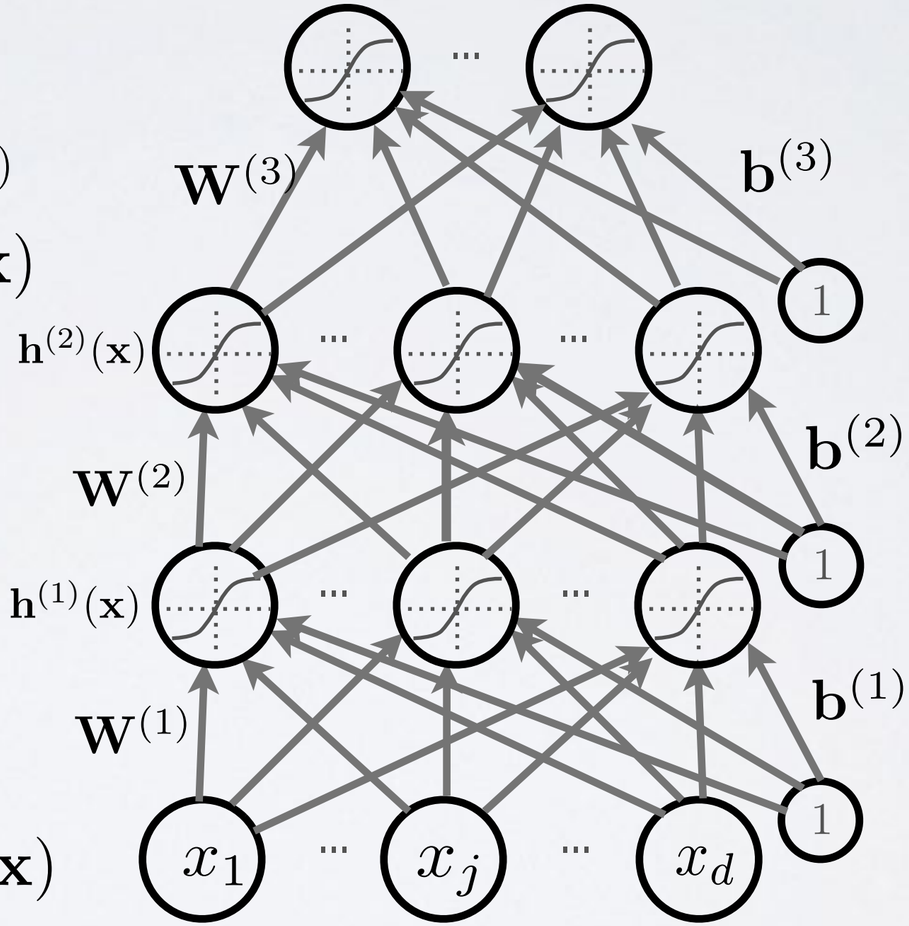
$$\mathbf{a}^{(k)}(\mathbf{x}) = \mathbf{b}^{(k)} + \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}(\mathbf{x})$$

  - hidden layer activation (k from 1 to L):

$$\mathbf{h}^{(k)}(\mathbf{x}) = \mathbf{g}(\mathbf{a}^{(k)}(\mathbf{x}))$$

  - output layer activation ($k=L+1$):

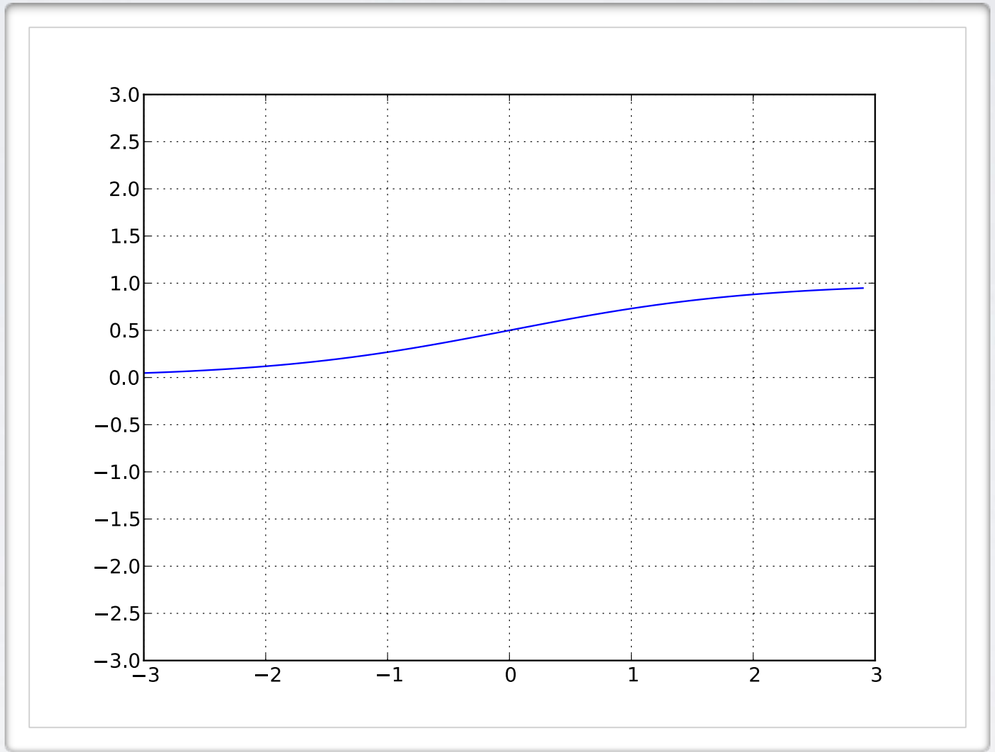
$$\mathbf{h}^{(L+1)}(\mathbf{x}) = \mathbf{o}(\mathbf{a}^{(L+1)}(\mathbf{x})) = \mathbf{f}(\mathbf{x})$$



ACTIVATION FUNCTION

Topics: sigmoid activation function

- Squashes the neuron's pre-activation between 0 and 1
- Always positive
- Bounded
- Strictly increasing

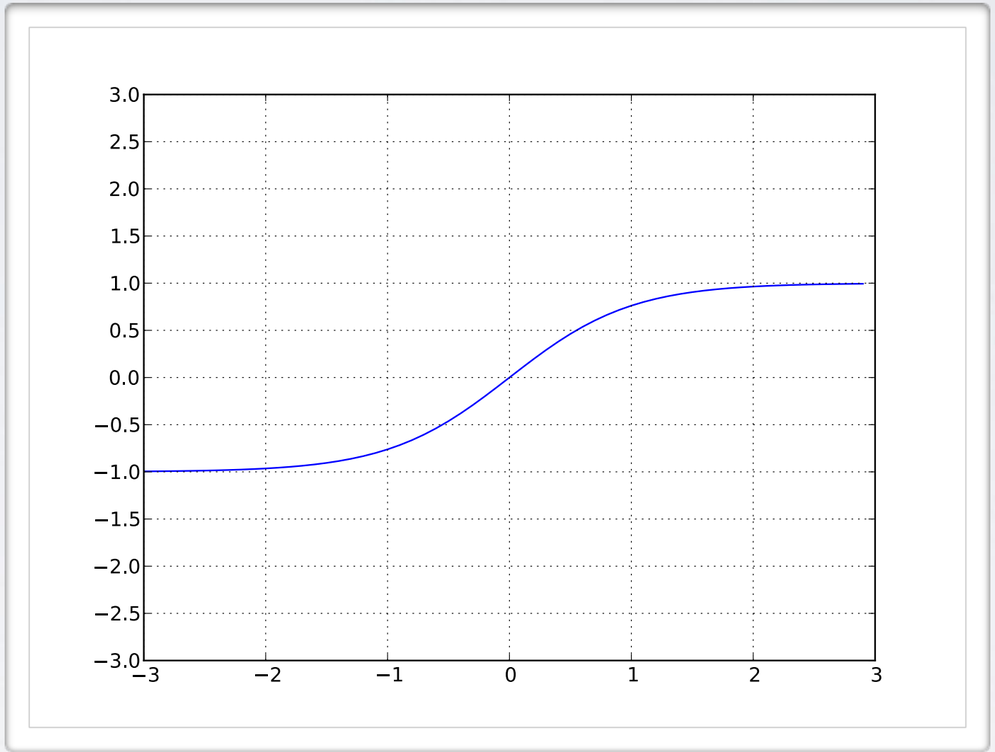


$$g(a) = \text{sigm}(a) = \frac{1}{1+\exp(-a)}$$

ACTIVATION FUNCTION

Topics: hyperbolic tangent ("tanh") activation function

- Squashes the neuron's pre-activation between -1 and 1
- Can be positive or negative
- Bounded
- Strictly increasing

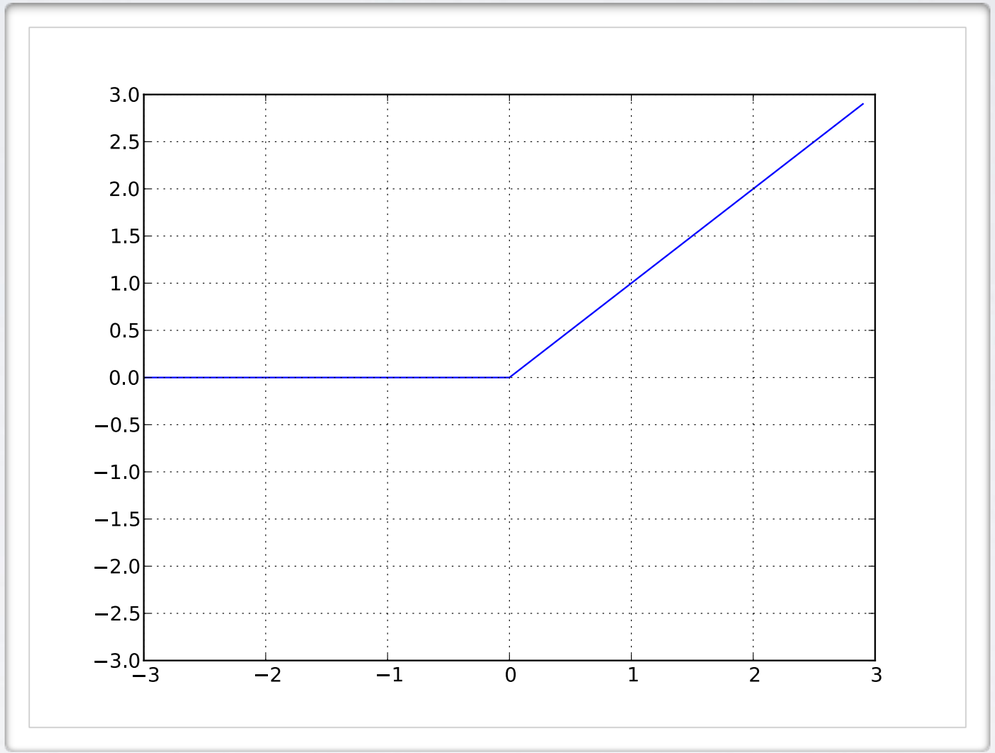


$$g(a) = \tanh(a) = \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)} = \frac{\exp(2a) - 1}{\exp(2a) + 1}$$

ACTIVATION FUNCTION

Topics: rectified linear activation function

- Bounded below by 0 (always non-negative)
- Not upper bounded
- Strictly increasing
- Tends to give neurons with sparse activities



$$g(a) = \text{reclin}(a) = \max(0, a)$$

ACTIVATION FUNCTION

Topics: softmax activation function

- For multi-class classification:
  - we need multiple outputs (1 output per class)
  - we would like to estimate the conditional probability $p(y = c|\mathbf{x})$

- We use the softmax activation function at the output:

$$\mathbf{o}(\mathbf{a}) = \text{softmax}(\mathbf{a}) = \left[\frac{\exp(a_1)}{\sum_c \exp(a_c)} \cdots \frac{\exp(a_C)}{\sum_c \exp(a_c)}\right]^\top$$

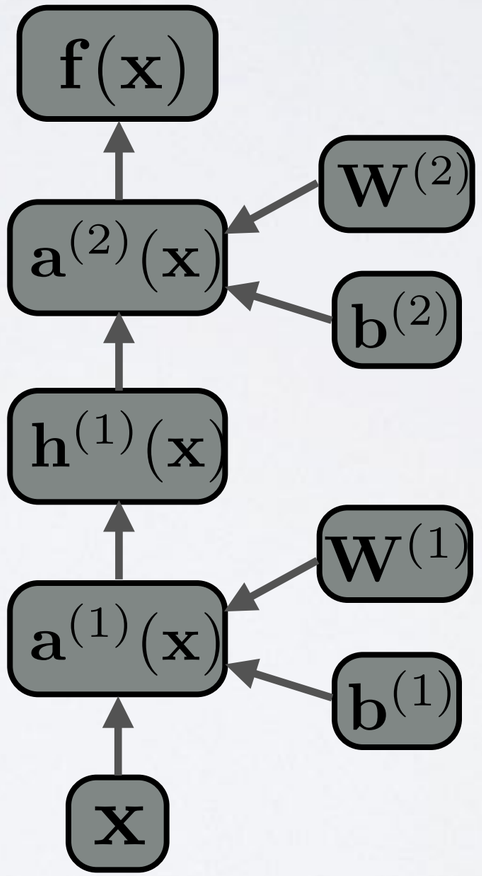
  - strictly positive
  - sums to one

- Predicted class is the one with highest estimated probability

FLOW GRAPH

Topics: flow graph

- Forward propagation can be represented as an acyclic flow graph
- It's a nice way of implementing forward propagation in a modular way
  - each box could be an object with an fprop method, that computes the value of the box given its parents
  - calling the fprop method of each box in the right order yield forward propagation



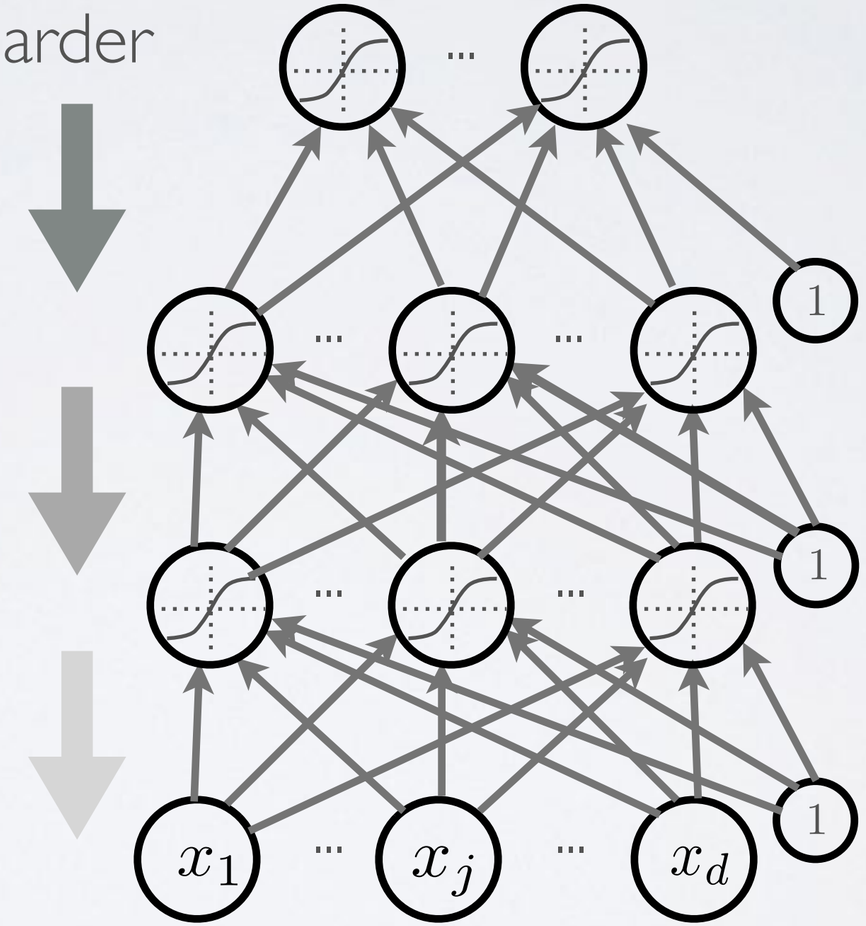
Neural Networks

Training deep feed-forward neural networks

DEEP LEARNING

Topics: why training is hard

- First hypothesis: optimization is harder (underfitting)
  - vanishing gradient problem
  - saturated units block gradient propagation

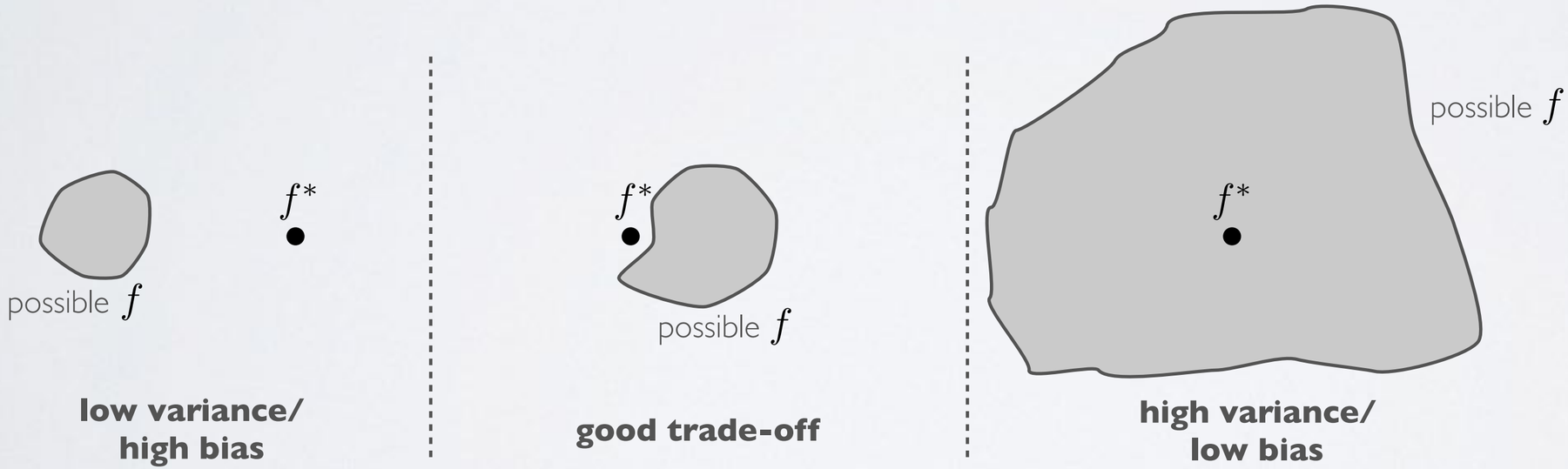


- This is a well known problem in recurrent neural networks

DEEP LEARNING

Topics: why training is hard

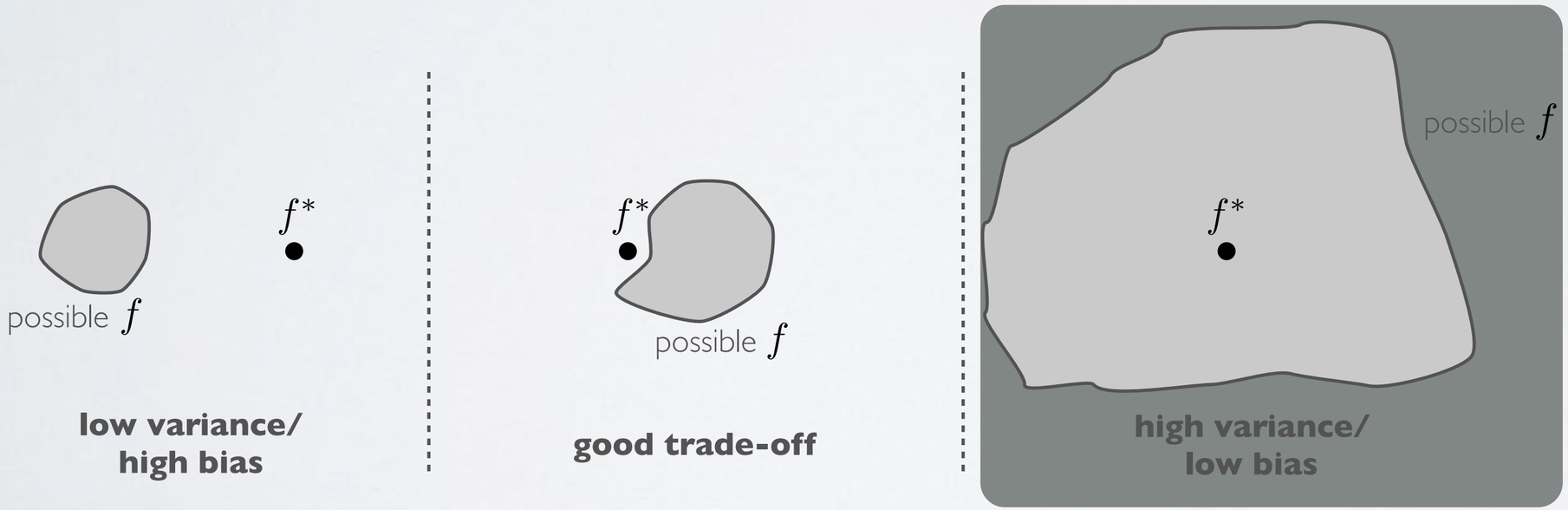
- Second hypothesis: overfitting
  - we are exploring a space of complex functions
  - deep nets usually have lots of parameters
- Might be in a high variance / low bias situation



DEEP LEARNING

Topics: why training is hard

- Second hypothesis: overfitting
  - we are exploring a space of complex functions
  - deep nets usually have lots of parameters
- Might be in a high variance / low bias situation



DEEP LEARNING

Topics: why training is hard

- Depending on the problem, one or the other situation will tend to dominate

- If first hypothesis (underfitting): better optimize
  - use better optimization methods
  - use GPUs

- If second hypothesis (overfitting): use better regularization
  - unsupervised pre-training
  - stochastic «dropout» training

DEEP LEARNING

Topics: why training is hard

- Depending on the problem, one or the other situation will tend to dominate

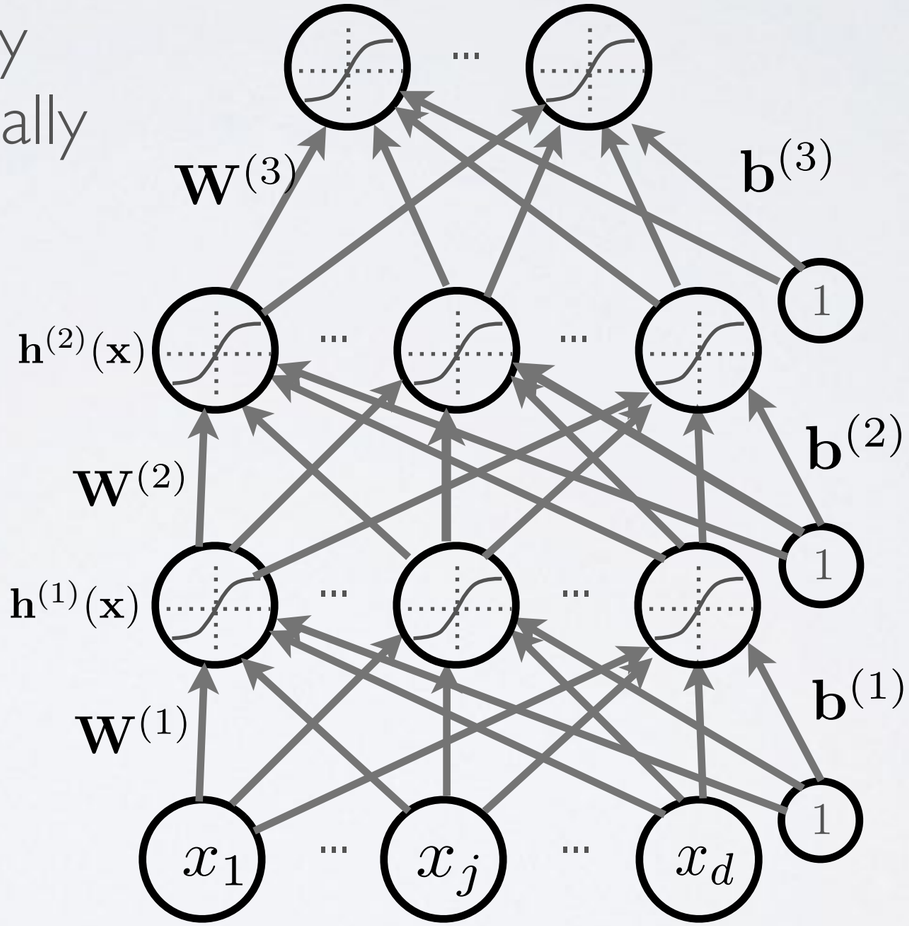
- If first hypothesis (underfitting): better optimize
  - use better optimization methods
  - use GPUs

- If second hypothesis (overfitting): use better regularization
  - **unsupervised pre-training**
  - stochastic «dropout» training

DROPOUT

Topics: dropout

- Idea: «cripple» neural network by removing hidden units stochastically
  - each hidden unit is set to 0 with probability 0.5
  - hidden units cannot co-adapt to other units
  - hidden units must be more generally useful

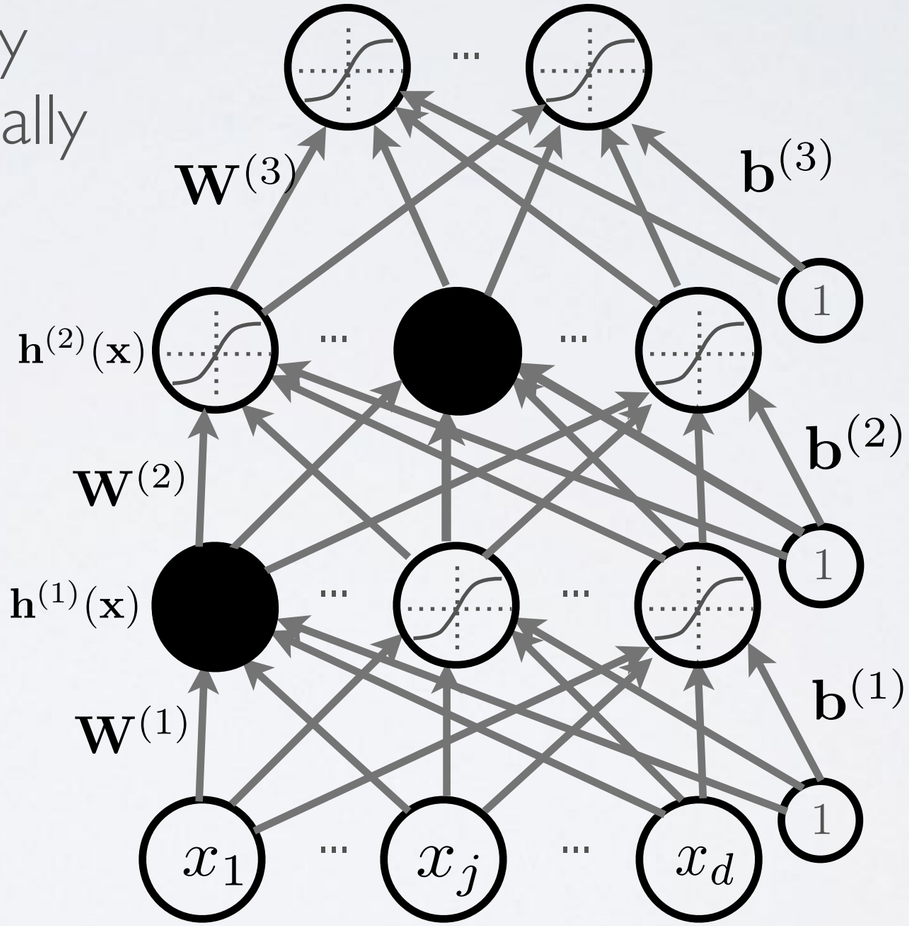


- Could use a different dropout probability, but 0.5 usually works well

DROPOUT

Topics: dropout

- Idea: «cripple» neural network by removing hidden units stochastically
  - each hidden unit is set to 0 with probability 0.5
  - hidden units cannot co-adapt to other units
  - hidden units must be more generally useful

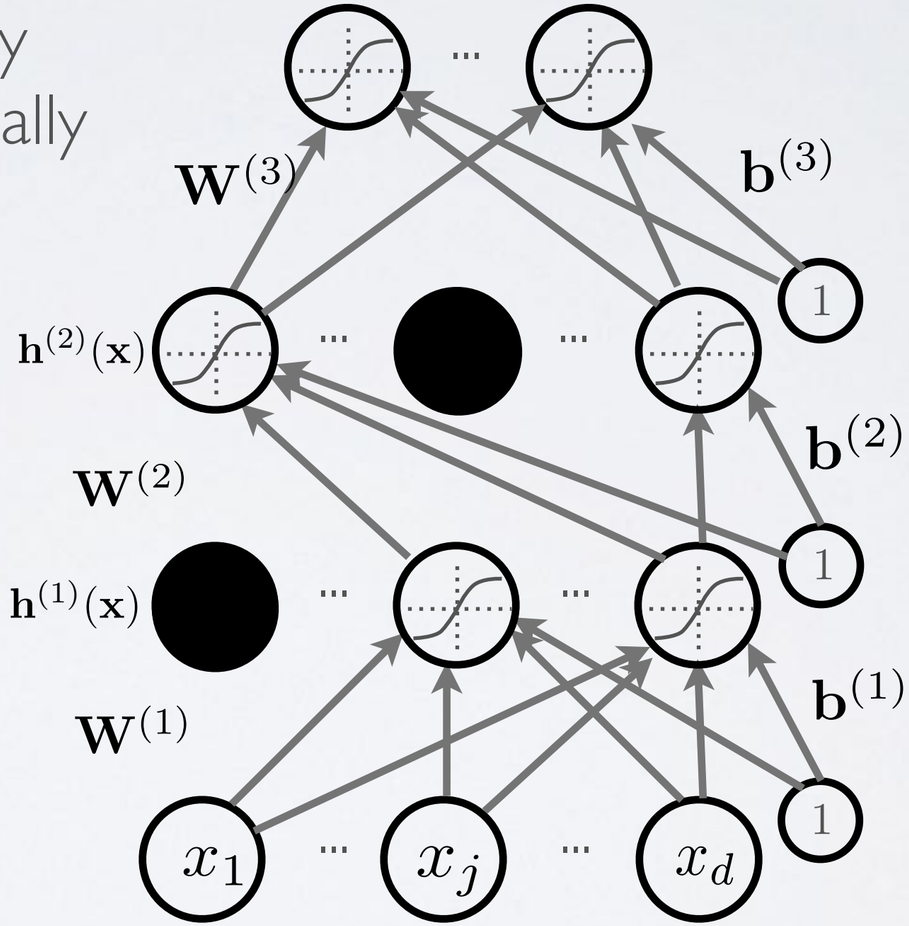


- Could use a different dropout probability, but 0.5 usually works well

DROPOUT

Topics: dropout

- Idea: «cripple» neural network by removing hidden units stochastically
  - each hidden unit is set to 0 with probability 0.5
  - hidden units cannot co-adapt to other units
  - hidden units must be more generally useful



- Could use a different dropout probability, but 0.5 usually works well

DROPOUT

Topics: dropout

- Use random binary masks $\mathbf{m}^{(k)}$
  - layer pre-activation for $k>0$ $(\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x})$

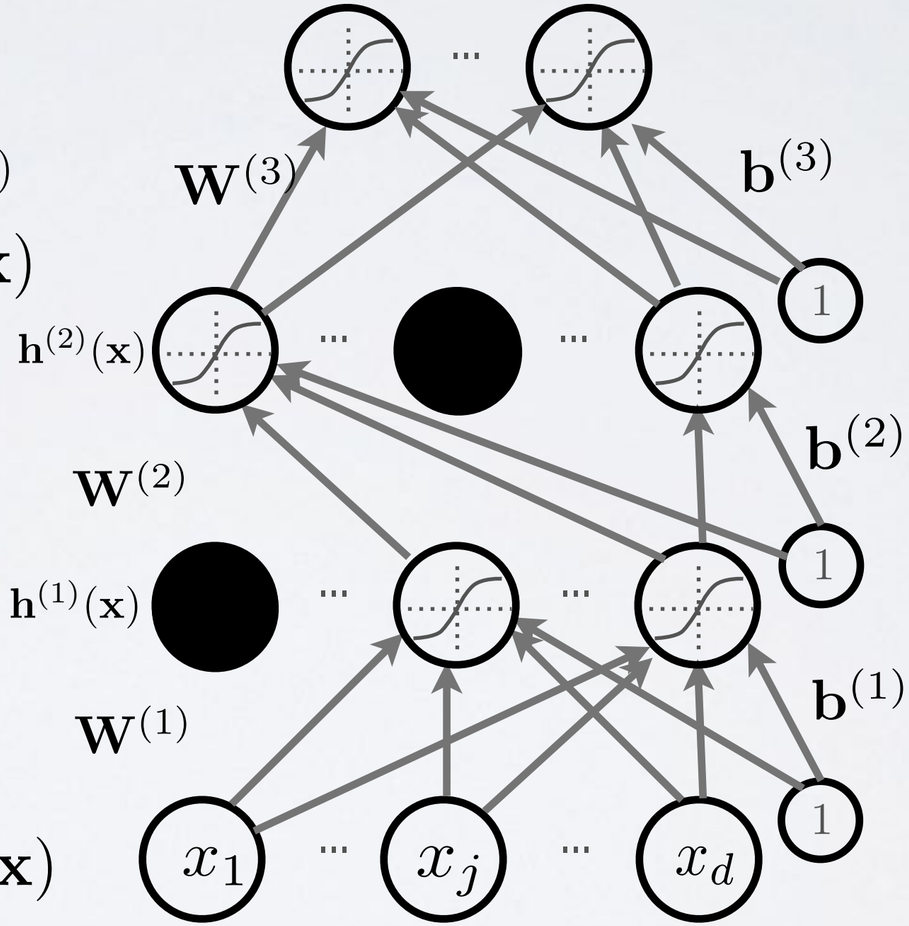
$$\mathbf{a}^{(k)}(\mathbf{x}) = \mathbf{b}^{(k)} + \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}(\mathbf{x})$$

  - hidden layer activation (k from 1 to L):

$$\mathbf{h}^{(k)}(\mathbf{x}) = \mathbf{g}(\mathbf{a}^{(k)}(\mathbf{x}))$$

  - output layer activation ($k=L+1$):

$$\mathbf{h}^{(L+1)}(\mathbf{x}) = \mathbf{o}(\mathbf{a}^{(L+1)}(\mathbf{x})) = \mathbf{f}(\mathbf{x})$$



DROPOUT

Topics: dropout

- Use random binary masks $\mathbf{m}^{(k)}$
  - layer pre-activation for $k>0$ $(\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x})$

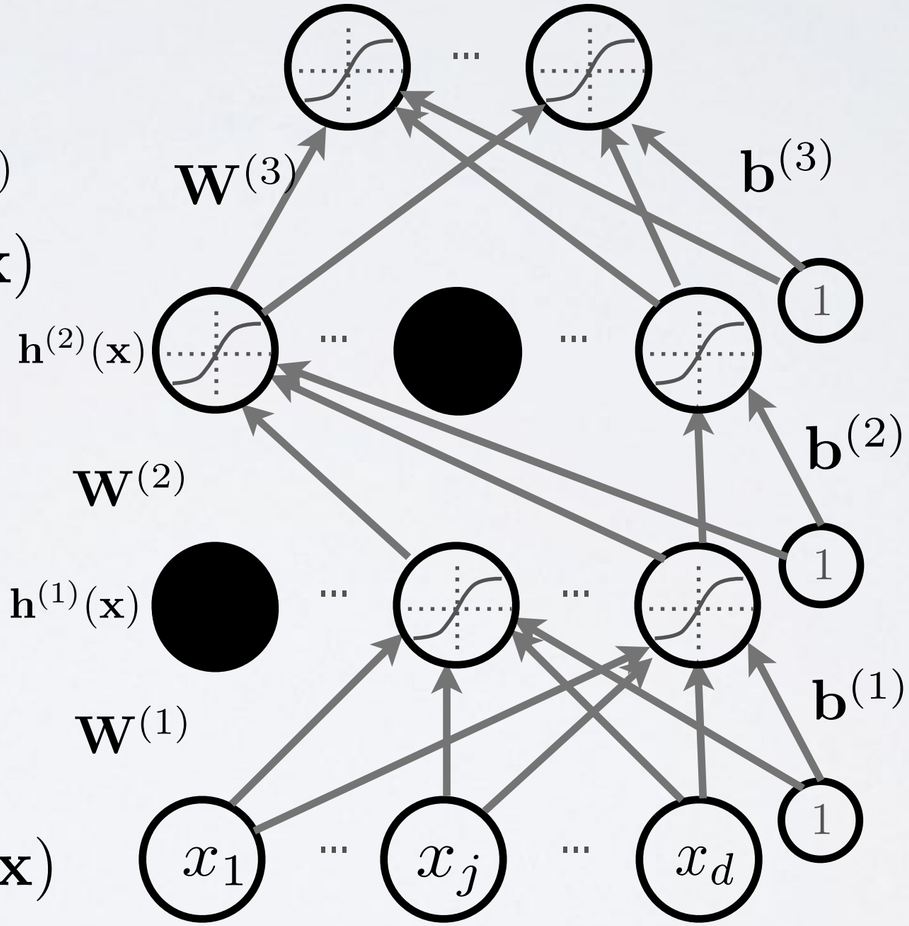
$$\mathbf{a}^{(k)}(\mathbf{x}) = \mathbf{b}^{(k)} + \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}(\mathbf{x})$$

  - hidden layer activation (k from 1 to L):

$$\mathbf{h}^{(k)}(\mathbf{x}) = \mathbf{g}(\mathbf{a}^{(k)}(\mathbf{x})) \odot \mathbf{m}^{(k)}$$

  - output layer activation ($k=L+1$):

$$\mathbf{h}^{(L+1)}(\mathbf{x}) = \mathbf{o}(\mathbf{a}^{(L+1)}(\mathbf{x})) = \mathbf{f}(\mathbf{x})$$



DEEP LEARNING

Topics: why training is hard

- Depending on the problem, one or the other situation will tend to dominate

- If first hypothesis (underfitting): better optimize
  - use better optimization methods
  - use GPUs

- If second hypothesis (overfitting): use better regularization
  - unsupervised pre-training
  - stochastic «dropout» training

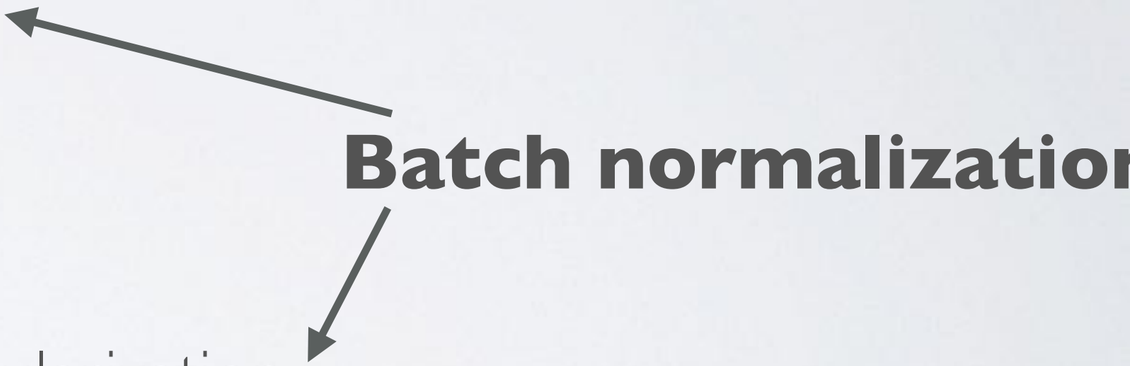
DEEP LEARNING

Topics: why training is hard

- Depending on the problem, one or the other situation will tend to dominate

- If first hypothesis (underfitting): better optimize
  - use better optimization methods
  - use GPUs

- If second hypothesis (overfitting): use better regularization
  - unsupervised pre-training
  - stochastic «dropout» training



BATCH NORMALIZATION

Topics: batch normalization

- Normalizing the inputs will speed up training
  (Lecun et al. 1998)
  - could normalization also be useful at the level of the hidden layers?

- **Batch normalization** is an attempt to do that
  (Ioffe and Szegedy, 2014)
  - each unit's **pre-**activation is normalized (mean subtraction, stddev division)
  - during training, mean and stddev is computed for **each minibatch**
  - backpropagation **takes into account** the normalization
  - at test time, the **global mean / stddev is used**

BATCH NORMALIZATION

Topics: batch normalization

- **Batch normalization**

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

BATCH NORMALIZATION

Topics: batch normalization

- **Batch normalization**

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Learned linear transformation to adapt to non-linear activation function

(γ and β are **trained**)

MERCI!