Machine Learning

Lecture 6 - Neural networks and Deep learning

Mattias Villani

Department of Statistics
Stockholm University

Department of Computer and Information Science
Linköping University


mattiasvillani.com


@matvil


mattiasvillani

Lecture overview

- Neural networks
- Deep learning

Deep neural networks

- **Nonlinear generalized linear model**

$$y|x_1, \ldots, x_p, \boldsymbol{\beta} \sim \text{ExpFamily}(\lambda)$$

$$g(\lambda) = f_\beta(x_1, \ldots, x_p)$$

  where $f_\beta(x_1, \ldots, x_p)$ can be linear or non-linear (spline or tree).
- Here: $f_\beta(x_1, \ldots, x_p)$ is a **neural network** with **hidden nodes**.
- **Richly parametrized models** that **learn the features**.
- (Huge) subculture in ML: frameworks and work pipelines.
- Their recent **success** comes from:
  - massive datasets
  - advances in parallel computing (graphic cards)
  - deep networks
  - better optimization algorithms (SGD etc)
  - automatic differentiation.

Neural network models

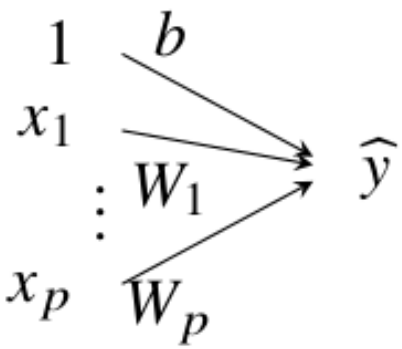
- Linear regression

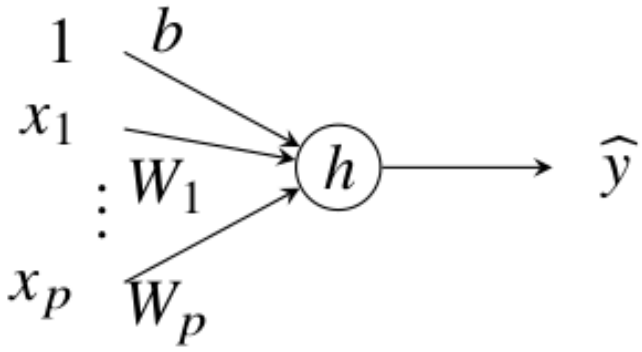
$$\hat{y} = W_1 x_1 + \ldots + W_p x_p + b$$

  - W_j are the **weights** (regression coefficients)
  - b is the **bias** (intercept). Very confusing terminology!

- **Activation function** $h : \mathbb{R} \to \mathbb{R}$ transforms to nonlinear

$$\hat{y} = h(W_1 x_1 + \ldots + W_p x_p + b)$$





Activation functions

- **Logistic** function (sigmoid)

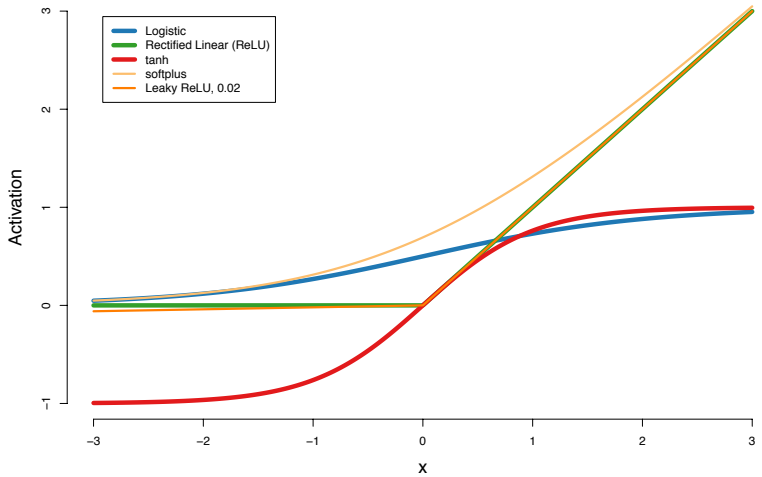
$$h(z) = \frac{1}{1 + e^{-z}}$$

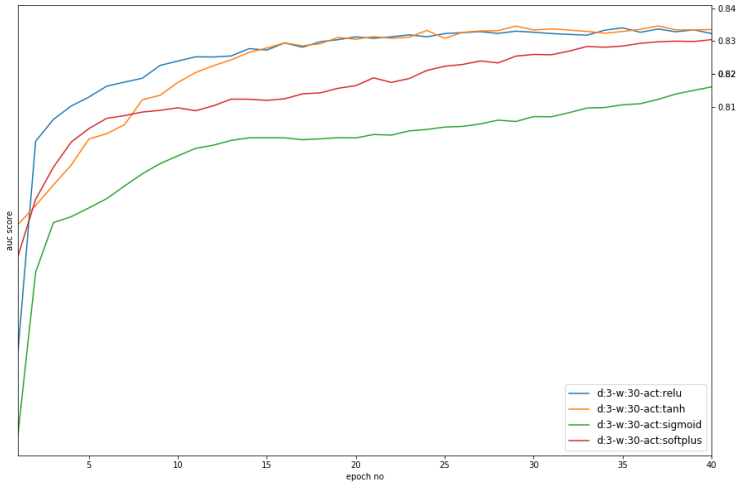
- Rectified linear (**ReLU**)

$$h(z) = \max(0, z)$$

- **Tanh**

$$h(z) = \max(0, z)$$



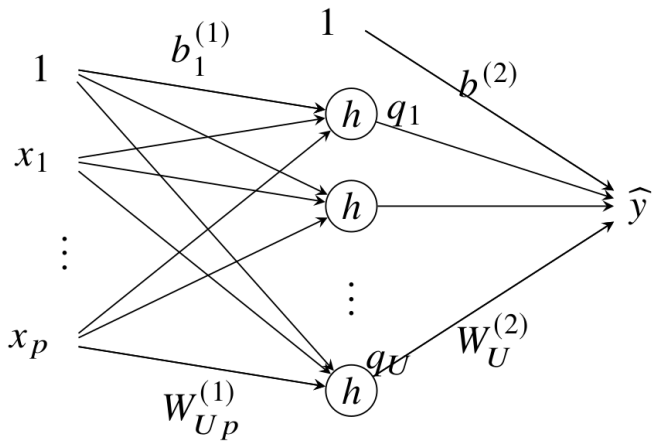


Two-layer neural network

- Add a layer of hidden units q between input x and output $\hat{y}$.

$$q_1 = h\left(W_{11}^{(1)} x_1 + \ldots + W_{1p}^{(1)} x_p + b_1^{(1)}\right)$$
$$q_2 = h\left(W_{21}^{(1)} x_1 + \ldots + W_{2p}^{(1)} x_p + b_2^{(1)}\right)$$
$$\vdots$$
$$q_U = h\left(W_{U1}^{(1)} x_1 + \ldots + W_{Up}^{(1)} x_p + b_U^{(1)}\right)$$
$$\hat{y} = W_1^{(2)} q_1 + \ldots + W_U^{(2)} q_U + b^{(2)}$$

Input variables Hidden units Output

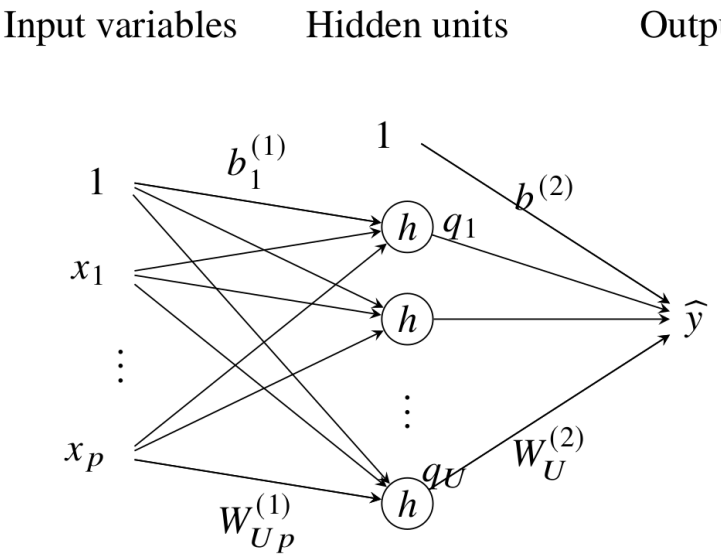


Two-layer neural network - vector form

- Vector form

$$\mathsf{q} = h\left(\mathsf{W}^{(1)}\mathsf{x} + \mathsf{b}^{(1)}\right)$$

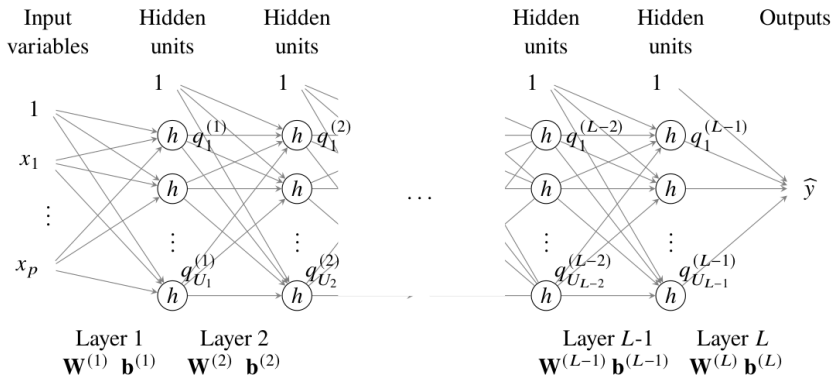
$$\hat{y} = \mathsf{W}^{(2)}\mathsf{q} + b^{(2)}$$



- **Hidden units**: $\mathsf{q} = (q_1, \ldots, q_U)^\top$.
- **Parameters**: $\boldsymbol{\theta} = \left(\mathrm{vec}(\mathsf{W}^{(1)})^\top, \mathsf{b}^{(1)\top}, \mathrm{vec}(\mathsf{W}^{(2)})^\top, b^{(2)}\right)^T$.

Deep neural network with L layers for regression

$$\mathsf{q}^{(1)} = h_1\left(\mathsf{W}^{(1)}\mathsf{x} + \mathsf{b}^{(1)}\right)$$
$$\mathsf{q}^{(2)} = h_2\left(\mathsf{W}^{(2)}\mathsf{q}^{(1)} + \mathsf{b}^{(2)}\right)$$
$$\vdots$$
$$\mathsf{q}^{(L-1)} = h_{L-1}\left(\mathsf{W}^{(L-1)}\mathsf{q}^{(L-2)} + \mathsf{b}^{(L-1)}\right)$$
$$\hat{y} = \mathsf{W}^{(L)}\mathsf{q}^{(L-1)} + b^{(L)}$$



- **Hidden units**: $\mathsf{q}^{(l)}$ is a U_l dimensional vector.
- **Parameters**: $\theta = \left(\mathrm{vec}(\mathsf{W}^{(1)})^\top, \mathsf{b}^{(1)\top}, \ldots, \mathrm{vec}(\mathsf{W}^{(L)})^\top, b^{(L)}\right)^T$.

Deep neural network for classification

- Deep neural network for **multi-class classification**

$$
\begin{aligned}
\mathsf{q}^{(1)} &= h_1\left(\mathsf{W}^{(1)}\mathsf{x} + \mathsf{b}^{(1)}\right) \\
\mathsf{q}^{(2)} &= h_2\left(\mathsf{W}^{(2)}\mathsf{q}^{(1)} + \mathsf{b}^{(2)}\right) \\
&\vdots \\
\mathsf{q}^{(L-1)} &= h_{L-1}\left(\mathsf{W}^{(L-1)}\mathsf{q}^{(L-2)} + \mathsf{b}^{(L-1)}\right) \\
\mathsf{z} &= \mathsf{W}^{(L)}\mathsf{q}^{(L-1)} + \mathsf{b}^{(L)} \\
\mathsf{g} &= \mathrm{softmax}(\mathsf{z})
\end{aligned}
$$

- **Softmax function** turns $\mathsf{z} \in \mathbb{R}^M$ into a probability distribution

$$
\mathrm{softmax}(\mathsf{z}) = \left(\frac{e^{z_1}}{\sum_{j=1}^{M} e^{z_j}}, \ldots, \frac{e^{z_M}}{\sum_{j=1}^{M} e^{z_j}}\right)^\top.
$$

Deep learning

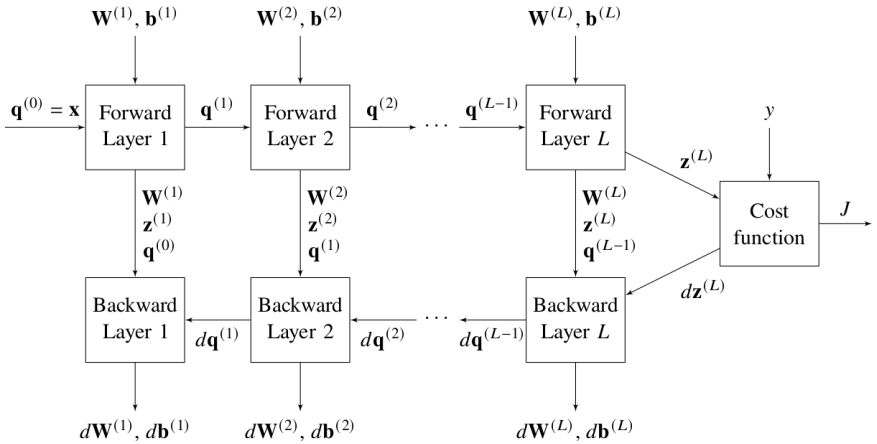
- Deep neural networks are typically learned by **optimization** methods.
- Challenge 1 - n **is large**.
  - Solution: stochastic gradient descent.
- Challenge 2 - $\dim(\theta)$ **is large**.
  - Solution: back-propagation and automatic differentiation.
- Challenge 3 - nonlinear with **many local minima** in objective.
  - Solution: stochastic gradient descent to escape local minima and automatic differentiation.

Backpropagation

- Computionally fast way to compute for $l = 1, \ldots, L$:
  - $\nabla_{W^{(l)}} J(W, b)$
  - $\nabla_{b^{(l)}} J(W, b)$

Backpropagation = Chain rule derivative + computational graph

- **Backprop**:
  - forward pass from x to $J(W, b)$ via all hidden $z^{(l)}$ and $q^{(l)}$.
  - backward pass from $J(W, b)$ to x to compute all derivatives.



Backpropagation for a 3-layer network

- Objective function

$$q^{(1)} = h_1\left(\underbrace{W^{(1)}x + b^{(1)}}_{z^{(1)}}\right)$$

$$q^{(2)} = h_2\left(\underbrace{W^{(2)}q^{(1)} + b^{(2)}}_{z^{(2)}}\right)$$

$$\hat{y} = \underbrace{W^{(3)}q^{(2)} + b^{(3)}}_{z^{(3)}}$$

$$J(\theta) = (\hat{y} - y)^2$$

- Some derivatives

$$\frac{\partial J(\theta)}{\partial b^{(1)}} = \frac{\partial J(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial q^{(2)}} \frac{\partial q^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial q^{(1)}} \frac{\partial q^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial b^{(1)}}$$

$$= 2(\hat{y} - y) \cdot 1 \cdot W^{(3)} h'(z^{(2)}) W^{(2)} h'(z^{(1)}) \cdot 1$$

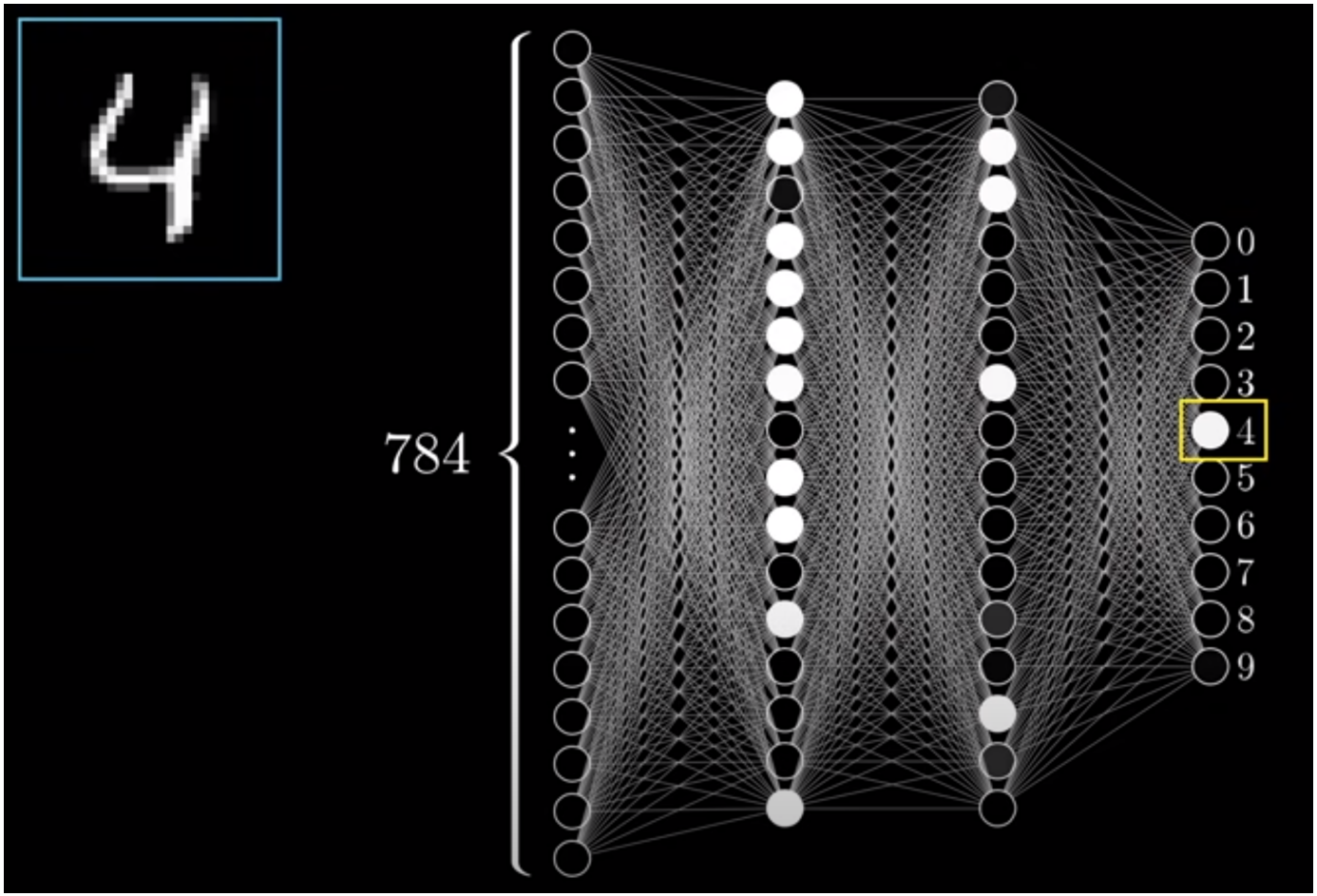
$$\frac{\partial J(\theta)}{\partial W^{(2)}} = \frac{\partial J(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial q^{(2)}} \frac{\partial q^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial W^{(2)}}$$

$$= 2(\hat{y} - y) \cdot 1 \cdot W^{(3)} h'(z^{(2)}) q^{(1)}$$

Demo of deep learning in R

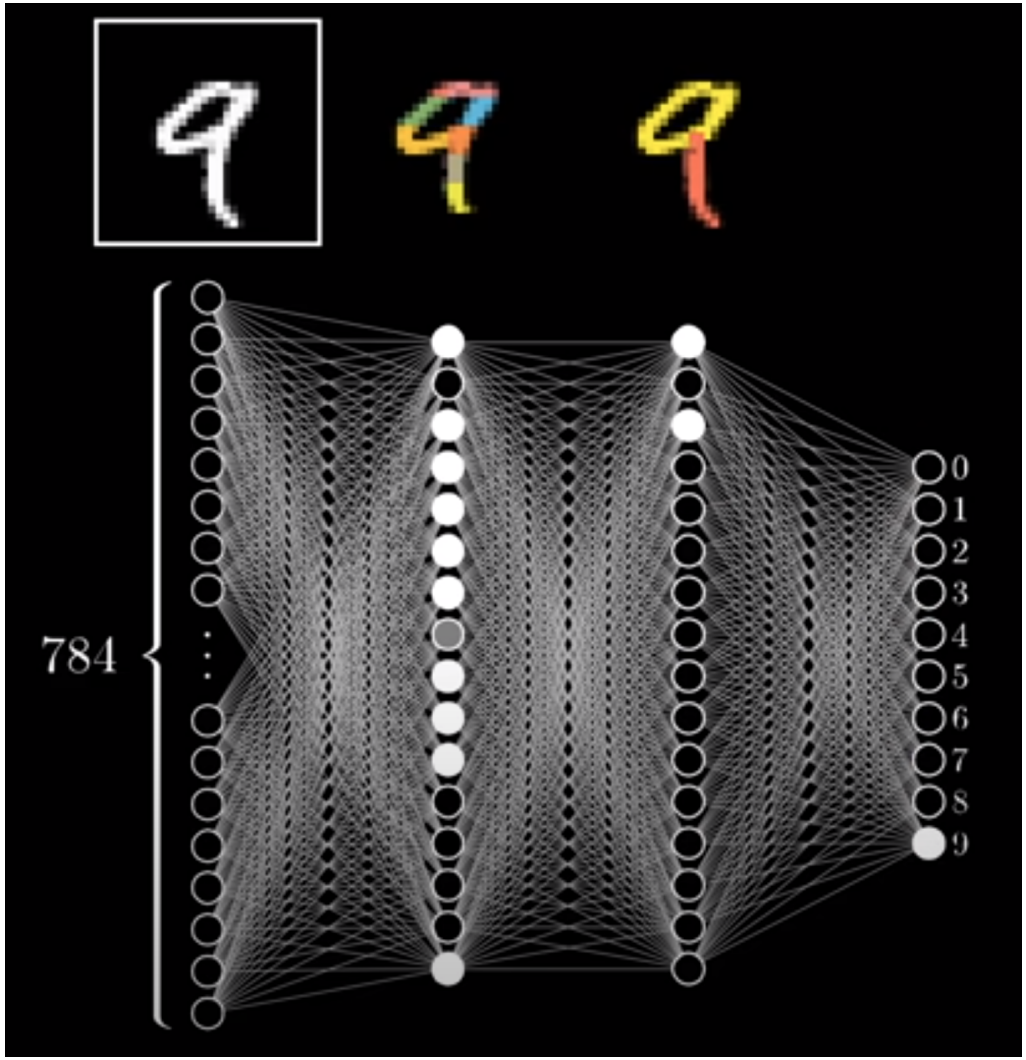
- MNIST dataset - the "hello world" of machine learning.
- The code is posted under this lecture.
- Experiment yourself with the code.
- Watch the video by the excellent Youtuber 3Blue1Brown

How 28×28 images propagates through the net



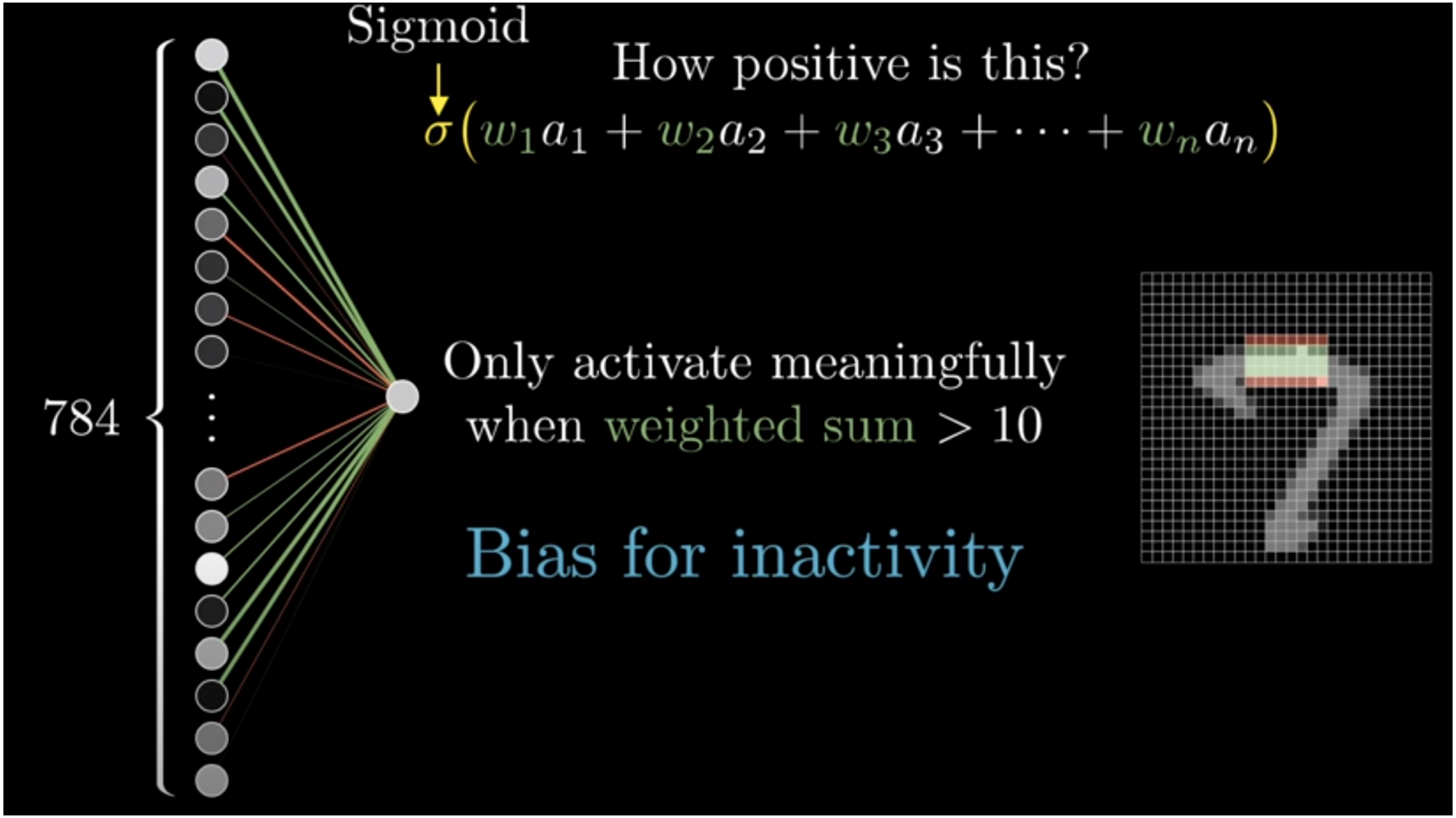
From video by 3Blue1Brown: https://youtu.be/aircAruvnKk

How a neural net learns the right features



From video by 3Blue1Brown: https://youtu.be/aircAruvnKk

How a neural net represents a edge



From video by 3Blue1Brown: https://youtu.be/aircAruvnKk