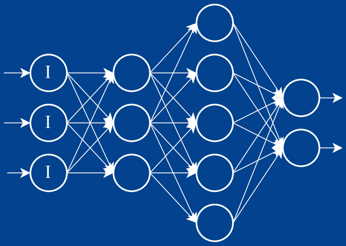
AARHUS
UNIVERSITY
SCHOOL OF ENGINEERING

MACHINE LEARNING

LESSON 9: Deep Learning I

CARSTEN EIE FRIGAARD

SPRING 2019



L09: Deep Learing I: Agenda

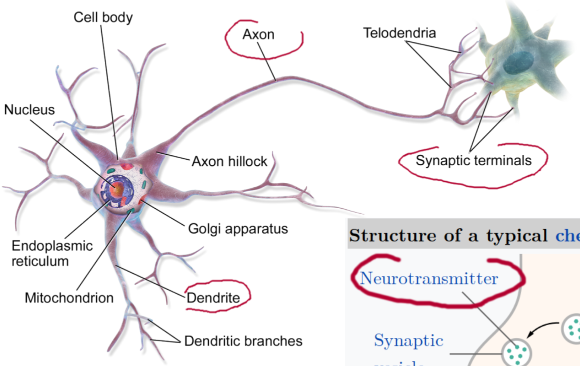
- BB | Hand-ins Deadlines: J2, 9 April.
- J3 custom projects: email a J3-project-description.
- GPU Cluster: use it, but remember GPU mem trick!

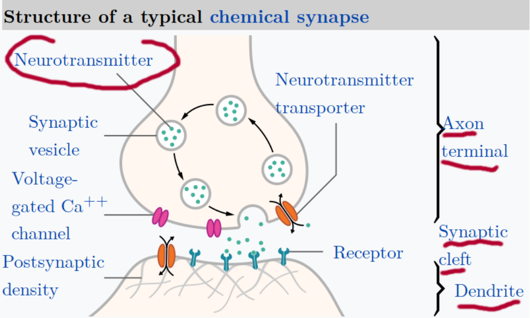
- The Biological Neuron,
  - historical points of neural nets,
  - GPU vs the brain,
  - exercise: `L09/neuron.ipynb` with ALT OPTIONAL exe.

- The Perceptron
  - activation functions
  - exercise: `L09/perceptron.ipynb`

- Multi-layer Perceptrons (MLPs)
  - definitions and more historical points
  - using Keras for building MLPs
  - exercise: `L09/keras_mlp_moon.ipynb`
  - exercise: `L09/keras_mlp_mnist.ipynb`

The Biological Neuron





Axons: nerve fibres connections to other neurons,

Dendrites: the input points to a neuron

Synapses: connection points btw. neurons.

[https://en.wikipedia.org/wiki/Axon]

History of Neural Networks

Engineered NN Systems Inspired by the Biological Brain

Focus has changed from biology understanding to beyond the neuroscientific perspective:

- began as engineered systems inspired by the biological brain—or as understanding of brain function,
- know as cybernetics in the 1940s–1960s,
- know as connectionism in the 1980s–1990s,
- now: *"a more general principle of learning [..], that are not necessarily neurally inspired"*, [DL].

Let's look at a history timeline of NNs..

History of Neural Networks

ML Summers and Winters [DL]

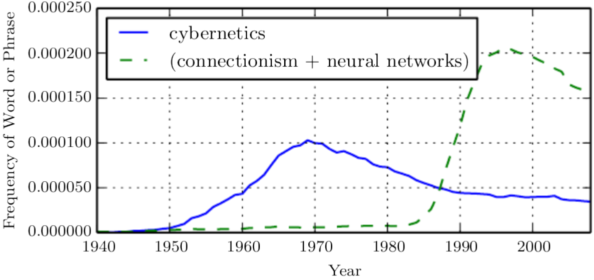


Figure 1.7: The figure shows two of the three historical waves of artificial neural nets research, as measured by the frequency of the phrases "cybernetics" and "connectionism" or "neural networks" according to Google Books (the third wave is too recent to appear). The first wave started with cybernetics in the 1940s–1960s, with the development of theories of biological learning (McCulloch and Pitts, 1943; Hebb, 1949) and implementations of the first models such as the perceptron (Rosenblatt, 1958) allowing the training of a single neuron. The second wave started with the connectionist approach of the 1980–1995 period, with back-propagation (Rumelhart *et al.*, 1986a) to train a neural network with one or two hidden layers. The current and third wave, deep learning, started around 2006 (Hinton *et al.*, 2006; Bengio *et al.*, 2007; Ranzato *et al.*, 2007a), and is just now appearing in book form as of 2016. The other two waves similarly appeared in book form much later than the corresponding scientific activity occurred.

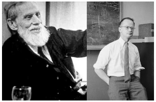
History of Neural Networks

Important Discoveries in the History of NN

- 1670: Leibniz, L’Hôpital, chain rule differentiation
- 1805/9: Legendre and Gauss, least-squares
- 1847: Cauchy, numerical gradient descent
- 1901: Pearson, PCA
- 1936: Fisher, linear discriminant analysis of iris data (probability theory: F-distribution)
- 1943: McCulloch and Pitts neuron
- 1949: Hebbs rule for self-organized learning
- 1957: Rosenblatts perceptron and supervised learning
- 1960-70: Multilayer perceptrons
- 1969: Minsky and Papert, XOR crisis, first winter
- 1985-86: BProp, LeCun, Rumelhart et at. second summer
- 95'ish: second winter
- 2006'ish: GPUs, start of third summer
- 2009: Jarrett, reLU activation function
- 2022'ish: third winter?

t

Mr. Fisher



Mr. McCulloch-Pitts

Mr. Rosenblatt

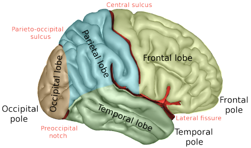
Mr. LeCun

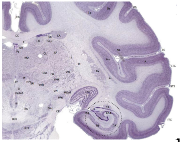
The GPU vs the Brain

GPU 1080		The Human Brain	
transistors	$7.2 \cdot 10^9$	neurons	$\sim 86 \cdot 10^9$
physical scale	16nm (FinFET)	physical scale	~ 0.05 mm
		synapses	$\sim 1.5 \cdot 10^{14}$
		connections per neuron	$\sim 10^3 - 10^4$
frequency	1.6 GHz	frequency	~ 10 Hz
power consumption	180 W (TPD)	power consumption	$\sim$ 13-20 W

Cerebral cortex

NOTE:
- African elephant neurons: $257 \cdot 10^9$
- Cerebral cortex neurons:
  - Human: $16 \cdot 10^9$
  - Long-finned pilot whale: $37 \cdot 10^9$





[https://hothardware.com/reviews/nvidia-geforce-gtx-1080-pascal-gpu-review]
[https://en.wikipedia.org/wiki/List_of_animals_by_number_of_neurons]
[https://en.wikipedia.org/wiki/Cerebral_cortex]
[https://hypertextbook.com/facts/2004/SamanthaCharles.shtml]

The Perceptron

Definition

History:

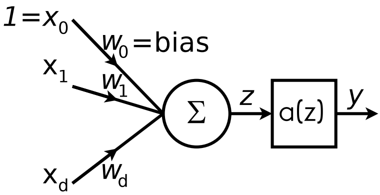
- 1943: McCulloch-Pitts: artificial neuron + network.
- 1957: Rosenblatt: the perceptron. Based on linear regressor + Heaviside activation function.

A linear regressor, with $\mathbf{x} \equiv [1 \ x_1 \ x_2 \ \cdots \ x_d]^\top$

$$
\begin{aligned}
z &= \mathbf{w}^\top \mathbf{x} \\
&= w_0 \cdot 1 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d
\end{aligned}
$$

plus activation function = the Perceptron (linear-threshold unit, LTU)

$$
y_{neuron}(\mathbf{x}; \mathbf{w}) = a(z) = a(\mathbf{w}^\top \mathbf{x})
$$



The Perceptron

The Activation Funtion, $a(\cdot)$

Original Rosenblatt perceptron activation fun

$$a_{\text{Rosenblatt}}(z) = \text{Heaviside}(z) = \begin{cases} 0, & \text{if } z < 0 \\ 1, & \text{if } z \geq 0 \end{cases}$$

but many other possible

$$a_{\text{sign}}(z) = \text{sgn}(z) = \begin{cases} -1, & \text{if } z < 0 \\ 0, & \text{if } z = 0 \\ 1, & \text{if } z \geq 0 \end{cases}$$

Much debated,

and different favorites thought different ML-*epochs*

...or *summers*.

The Perceptron

More Activation Funtions, $a(\cdot)$

Logistic Sigmoid

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

$$\frac{da(z)}{dx} = \frac{e^{-z}}{(1+e^{-z})^2}$$

$$= a(z)(1 - a(z))$$

Hyperbolic Tangens

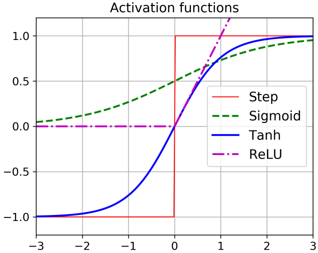
$$a(z) = \tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$$

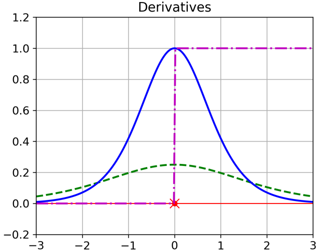
$$\frac{da(z)}{dx} = 1 - (\mathrm{a(z)})^2$$

Rectified Linear Unit, ReLU

$$a(z) = z^{+} = \max(0, z) = \begin{cases} z, & \text{for } z > 0 \\ 0, & \text{else} \end{cases}$$

$$\frac{da(z)}{dx} = \begin{cases} 1, & \text{for } z > 0 \\ 0, & \text{else} \end{cases}$$



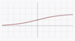


The Perceptron

Properties of the Activation Function

- linear: if multiple layers use identity $a(z) = z$, the network is equivalent to a single-layer,
- non-linear: two-layer neural network is a universal function approximator,
- (continuously) differentiable: for learning algo,
- monotonic: convex error surf for single layer model.

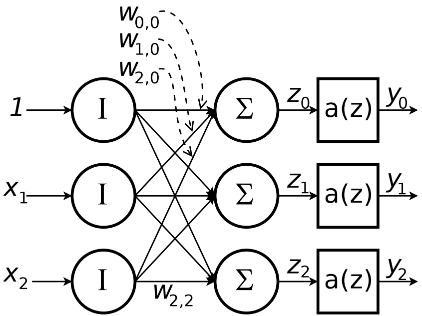
[https://en.wikipedia.org/wiki/Activation_function]

Name	Plot	Equation	Derivative (with respect to x)	Range	Order of continuity	Monotonic	Monotonic derivative
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$	C^∞	Yes	Yes
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$	C^{-1}	Yes	No
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$ [1]	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$	C^∞	Yes	No
TanH		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$	C^∞	Yes	No
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$	$\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$	C^∞	Yes	No
ArSinH		$f(x) = \sinh^{-1}(x) = \ln\left(x + \sqrt{x^2+1}\right)$	$f'(x) = \frac{1}{\sqrt{x^2+1}}$	$(-\infty, \infty)$	C^∞	Yes	No
ElliotSig[9][10][11]		[illegible]	[illegible]				

Multi-layer Perceptrons (MLPs)

MLP, but only one layer...

Stacking up neurons or perceptrons into an array

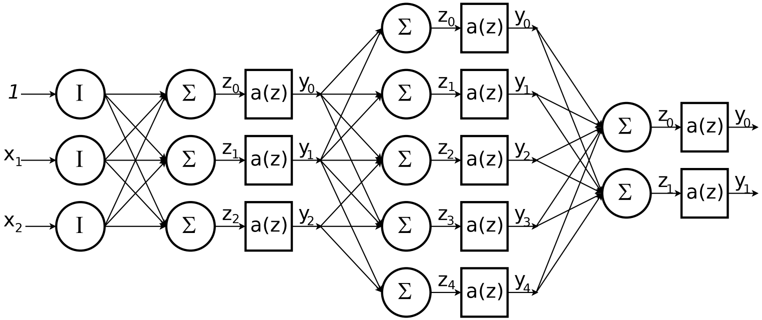


Deep feedforward networks, also often called feedforward neural networks, or multi-layer perceptrons (MLPs), are the quintessential deep learning models. [DL, p167]

Multi-layer Perceptrons (MLPs)

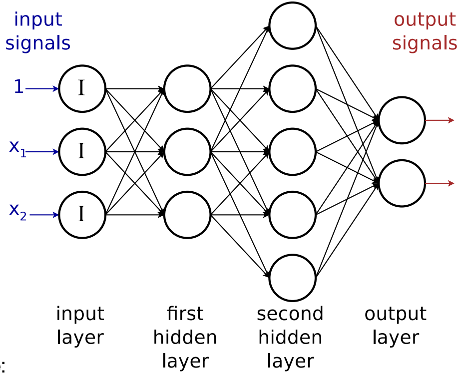
MLP, three layers, fully connected...

Stacking arrays of perceptrons into layers



Multi-layer Perceptrons (MLPs)

MLP, three layers, fully connected, simplified nodes...



MLP nomenclature:

input layer: handles the input **x** data,
output layer: only visible output signal from the network,
hidden layer(s): internal layers in the network,
fully connected: all nodes in layer connected to all neurons in the previous/next layer,
feed-forward: the signals flows only forward in the network (in contrast to feed-backward in BProp),
Backpropagation: or BProp, training algo of NN, more in later lesson,
Deep-learning networks: MLP with several hidden layers, say >2 ?

Multi-layer Perceptrons (MLPs)

From Perceptron training to MLP Training

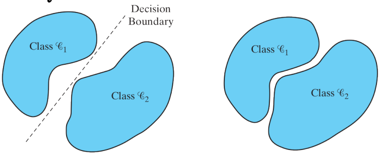
Training perceptrons and Hebb's postulate:

"Cells that fire together wire together."

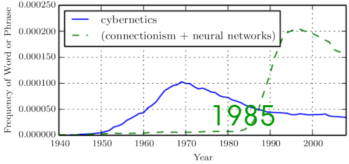
and

[..] explain synaptic plasticity, the adaptation of brain neurons during the learning process.

[https://en.wikipedia.org/wiki/Hebbian_theory]



- for linearly separable problems: perceptron convergence theorem,
- but what about XOR problems: Minsky/Papert XOR crisis,
- and how do you train MLP's?



No method existed until..

1985/86: LeCun, Rumelhart et al.: Backpropagation (L11)

Multi-layer Perceptrons (MLPs)

History: Dataset Size vs Time [DL]

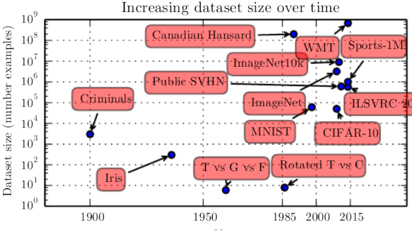


Figure 1.8: Dataset sizes have increased greatly over time. In the early 1900s, statisticians studied datasets using hundreds or thousands of manually compiled measurements (Garson, 1900; Gosset, 1908; Anderson, 1935; Fisher, 1936). In the 1950s through 1980s, the pioneers of biologically inspired machine learning often worked with small, synthetic datasets, such as low-resolution bitmaps of letters, that were designed to incur low computational cost and demonstrate that neural networks were able to learn specific kinds of functions (Widrow and Hoff, 1960; Rumelhart *et al.*, 1986b). In the 1980s and 1990s, machine learning became more statistical in nature and began to leverage larger datasets containing tens of thousands of examples such as the MNIST dataset (shown in Fig. 1.9) of scans of handwritten numbers (LeCun *et al.*, 1998b). In the first decade of the 2000s, more sophisticated datasets of this same size, such as the CIFAR-10 dataset (Krizhevsky and Hinton, 2009) continued to be produced. Toward the end of that decade and throughout the first half of the 2010s, significantly larger datasets, containing hundreds of thousands to tens of millions of examples, completely changed what was possible with deep learning. These datasets included the public Street View House Numbers dataset (Netzer *et al.*, 2011), various versions of the ImageNet dataset (Deng *et al.*, 2009, 2010a; Russakovsky *et al.*, 2014a), and the Sports-1M dataset (Karpathy *et al.*, 2014). At the top of the graph, we see that datasets of translated sentences, such as IBM's dataset constructed from the Canadian Hansard (Brown *et al.*, 1990) and the WMT 2014 English to French dataset (Schwenk, 2014) are typically far ahead of other dataset sizes.

Multi-layer Perceptrons (MLPs)

History: MLP Network Size vs Time [DL]

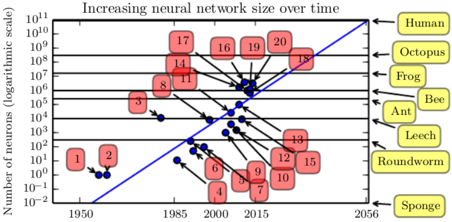


Figure 1.11: Since the introduction of hidden units, artificial neural networks have doubled in size roughly every 2.4 years. Biological neural network sizes from Wikipedia (2015).

1. Perceptron (Rosenblatt, 1958, 1962)
2. Adaptive linear element (Widrow and Hoff, 1960)
3. Neocognitron (Fukushima, 1980)
4. Early back-propagation network (Rumelhart *et al.*, 1986b)
5. Recurrent neural network for speech recognition (Robinson and Fallside, 1991)
6. Multilayer perceptron for speech recognition (Bengio *et al.*, 1991)
7. Mean field sigmoid belief network (Saul *et al.*, 1996)
8. LeNet-5 (LeCun *et al.*, 1998b)
9. Echo state network (Jaeger and Haas, 2004)
10. Deep belief network (Hinton *et al.*, 2006)
11. GPU-accelerated convolutional network (Chellapilla *et al.*, 2006)
12. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
13. GPU-accelerated deep belief network (Raina *et al.*, 2009)
14. Unsupervised convolutional network (Jarrett *et al.*, 2009)
15. GPU-accelerated multilayer perceptron (Ciresan *et al.*, 2010)
16. OMP-1 network (Coates and Ng, 2011)
17. Distributed autoencoder (Le *et al.*, 2012)
18. Multi-GPU convolutional network (Krizhevsky *et al.*, 2012)
19. COTS HPC unsupervised convolutional network (Coates *et al.*, 2013)
20. GoogLeNet (Szegedy *et al.*, 2014a)

Multi-layer Perceptrons (MLPs)

History: MLP Neuron Connections vs Time [DL]

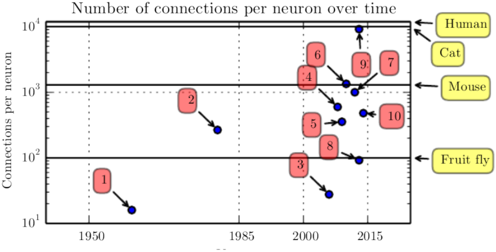


Figure 1.10: Initially, the number of connections between neurons in artificial neural networks was limited by hardware capabilities. Today, the number of connections between neurons is mostly a design consideration. Some artificial neural networks have nearly as many connections per neuron as a cat, and it is quite common for other neural networks to have as many connections per neuron as smaller mammals like mice. Even the human brain does not have an exorbitant amount of connections per neuron. Biological neural network sizes from Wikipedia (2015).

1. Adaptive linear element (Widrow and Hoff, 1960)
2. Neocognitron (Fukushima, 1980)
3. GPU-accelerated convolutional network (Chellapilla *et al.*, 2006)
4. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
5. Unsupervised convolutional network (Jarrett *et al.*, 2009)
6. GPU-accelerated multilayer perceptron (Ciresan *et al.*, 2010)
7. Distributed autoencoder (Le *et al.*, 2012)
8. Multi-GPU convolutional network (Krizhevsky *et al.*, 2012)
9. COTS HPC unsupervised convolutional network (Coates *et al.*, 2013)
10. GoogLeNet (Szegedy *et al.*, 2014a)

Multi-layer Perceptrons (MLPs)

History: MLP Error Rate vs Time [DL]

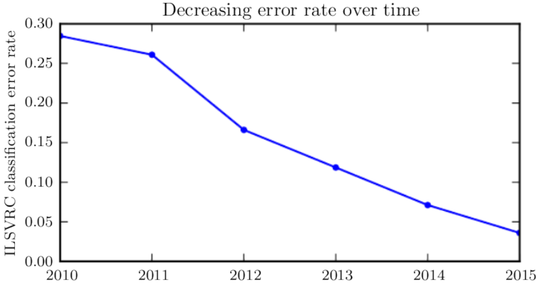
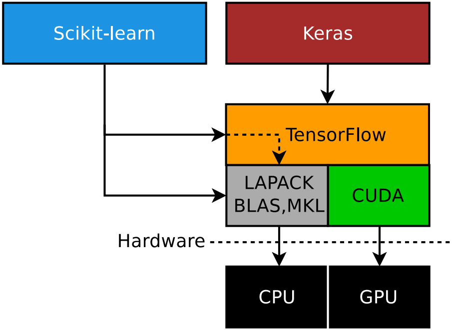
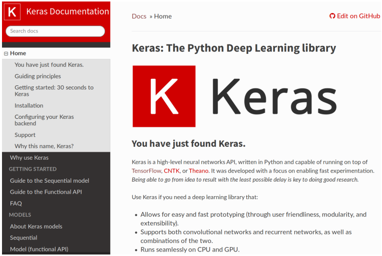


Figure 1.12: Since deep networks reached the scale necessary to compete in the ImageNet Large Scale Visual Recognition Challenge, they have consistently won the competition every year, and yielded lower and lower error rates each time. Data from Russakovsky *et al.* (2014b) and He *et al.* (2015).

Keras and Tensorflow

Using the Keras API instead of Scikit-learn or TensorFlow



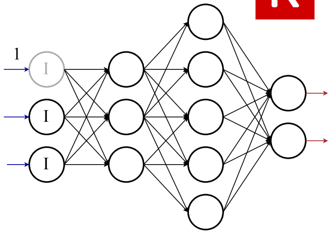


NOTE:

- documentation: `https://keras.io/`
- keras provides a `fit-predict`-interface,
- many similiarities to Scikit-learn,
- but also many differences!

Building Keras MLPs

Using the Keras `Sequential` class,
programatical build up model:




```
# Build Keras model
model = Sequential()
model.add(Dense(input_dim=2, units=3, activation="tanh", ..)
model.add(Dense(units=5, activation="relu", ..)
model.add(Dense(units=2, activation="softmax"))
.
.
X_train, .. = train_test_split(X, y, .. )
.
.
y_train_categorical = to_categorical(y_train, num_classes=2)
y_test_categorical  = to_categorical(y_test,  num_classes=2)
.
.
history = model.fit(X_train, y_train_categorical, ..
.
.
score = model.evaluate(X_test, y_test_categorical)
```

Input Layer: Categorical Encoding

For MLP Classification

One-hot `to_categorical(·)` encoding in Keras:

- input layer: one-hot class encoding,
- output layer: one neuron per output class that fires,
  and use `softmax` for output neurons
- beware of misformated classes.

```
import numpy as np
from keras.utils.np_utils import to_categorical

y = np.array([1, 2, 0, 4, -1])
y_cat = to_categorical(y)

print(y_cat)

#[[0. 1. 0. 0. 0.]  => i=0, class 1
# [0. 0. 1. 0. 0.]  => i=1, class 2
# [1. 0. 0. 0. 0.]  => i=2, class 0
# [0. 0. 0. 0. 1.]  => i=3, class 4
# [0. 0. 0. 0. 1.]] => i=4, also class 4!
#                      NOTE: no class 3
```

[L09/Extra/keres_to_categorical.ipynb]

Output Layer: Softmax Function

For MLP Classification: Assing a Probability for each Class

Softmax (softargmax/normalized exponential) definition

$$\mathrm{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{i=1}^{n} e^{x_i}}$$

`softmax`: smooth approx. of `argmax` function.

`argmax`: the index-of-the-max-value for some data.

```python
# python demo of softmax/argmax
x = np.array([1, 2, -4, 5, 1])
i = np.argmax(x)

PrintMatrix(x,"x = ")
print(f"np.argmax(x) = {np.argmax(x)}")

def softmax(x):
    z = np.exp(x)
    s = np.sum(z)
    return z / s

PrintMatrix(softmax(x),"softmax(x) = ")
print(f"np.argmax(softmax(x)) = {np.argmax(softmax(x))}")
```

```
# output
x = [ 1  2 -4  5  1 ]
np.argmax(x) = 3
softmax(x) = [0.02 0.05 0.  0.92 0.02]
np.argmax(softmax(x)) = 3
```

[L09/Extra/softmax.ipynb]

MLP Effect of Number of Hidden Layers

How Many Hidden Layers?

MNIST Search Quest Exercise:

- ITMAL Grp10:
  used a 20-50-70-100-70-50-20 layer `MLPClassifier`,
- found layers by trial-and-error,
- but what are the optimal hidden layer sizes and neurons per hidden layer?

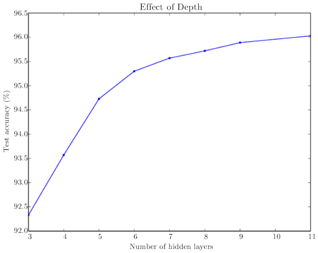


Figure 6.6: Empirical results showing that deeper networks generalize better when used to transcribe multi-digit numbers from photographs of addresses. Data from Goodfellow *et al.* (2014d). The test set accuracy consistently increases with increasing depth. See Fig. 6.7 for a control experiment demonstrating that other increases to the model size do not yield the same effect.