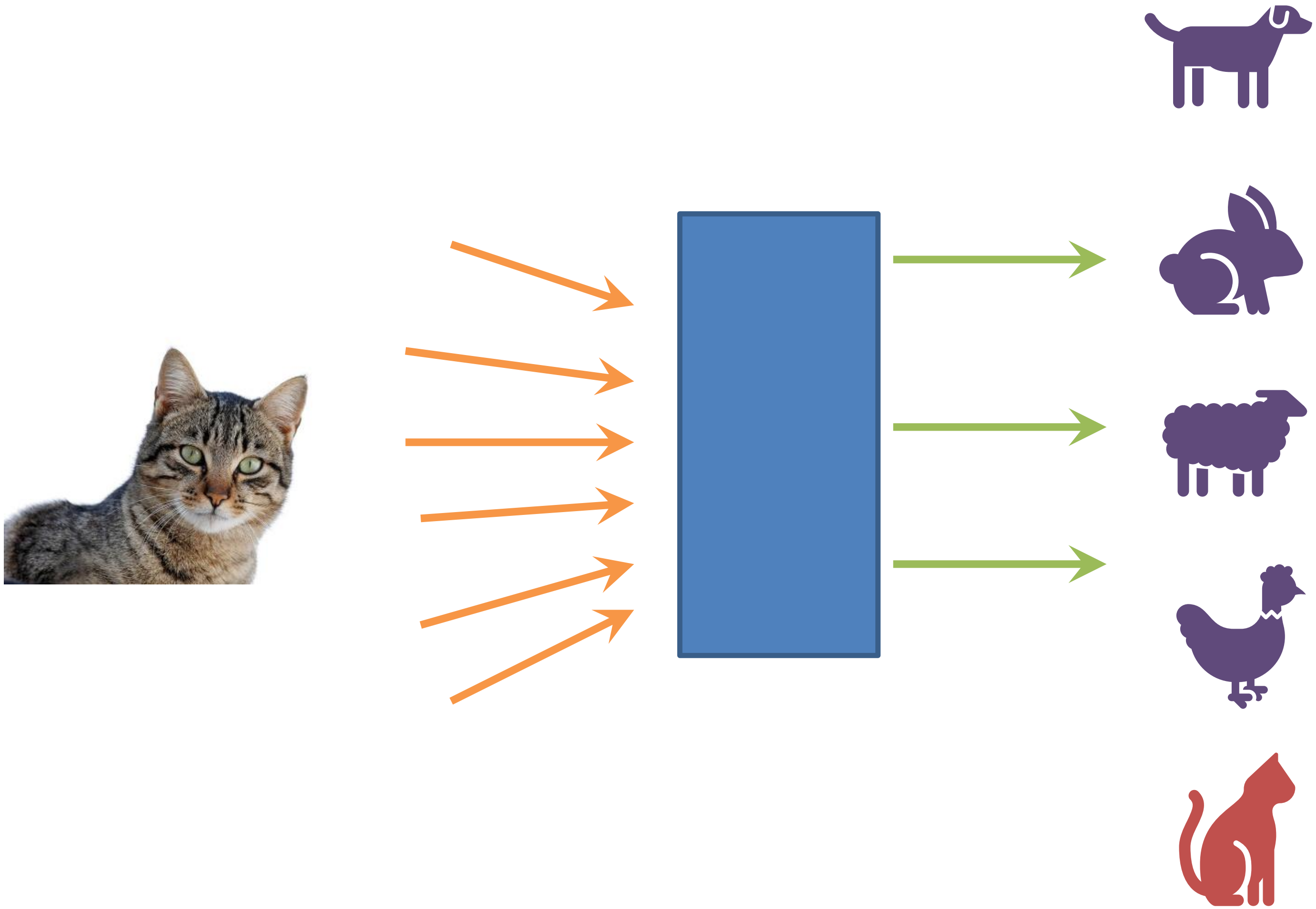
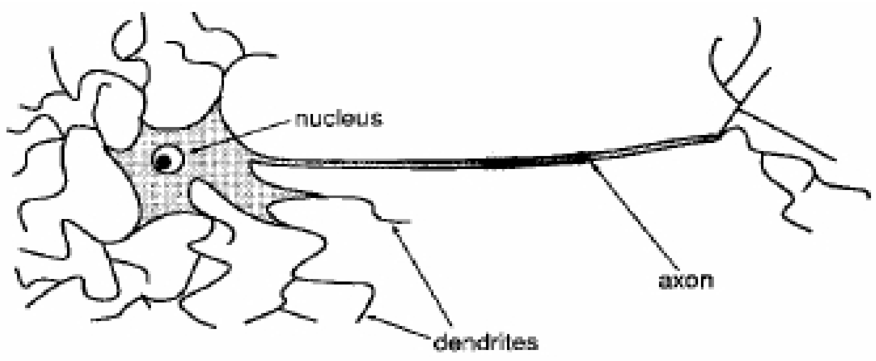
Neural Networks

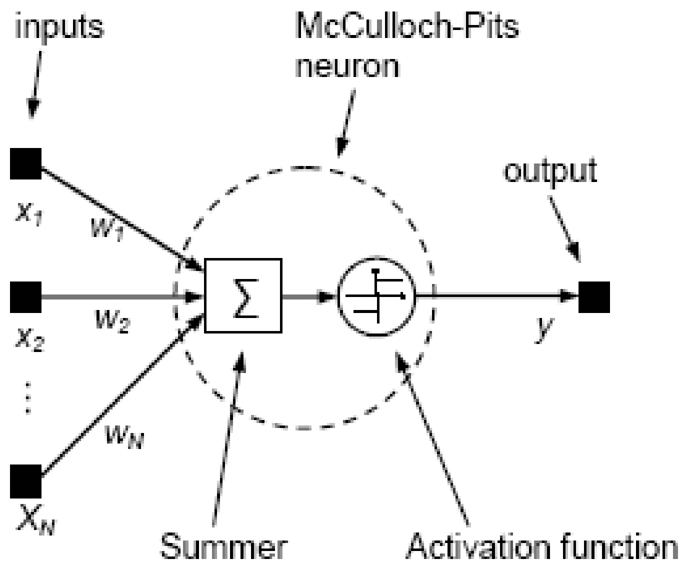
Classification Networks



https://commons.wikimedia.org/wiki/File:Cat_March_gpj.2010-1

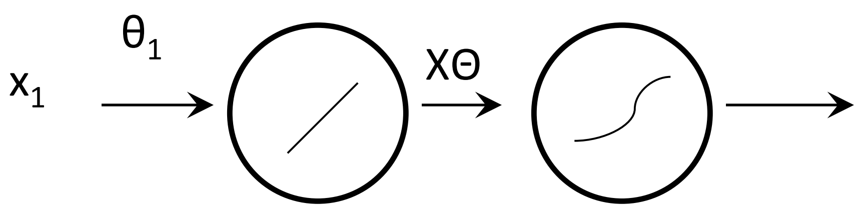
Warren McCulloch and Walter Pitts (1943)





http://www.controlglobal.com/articles//2006/221

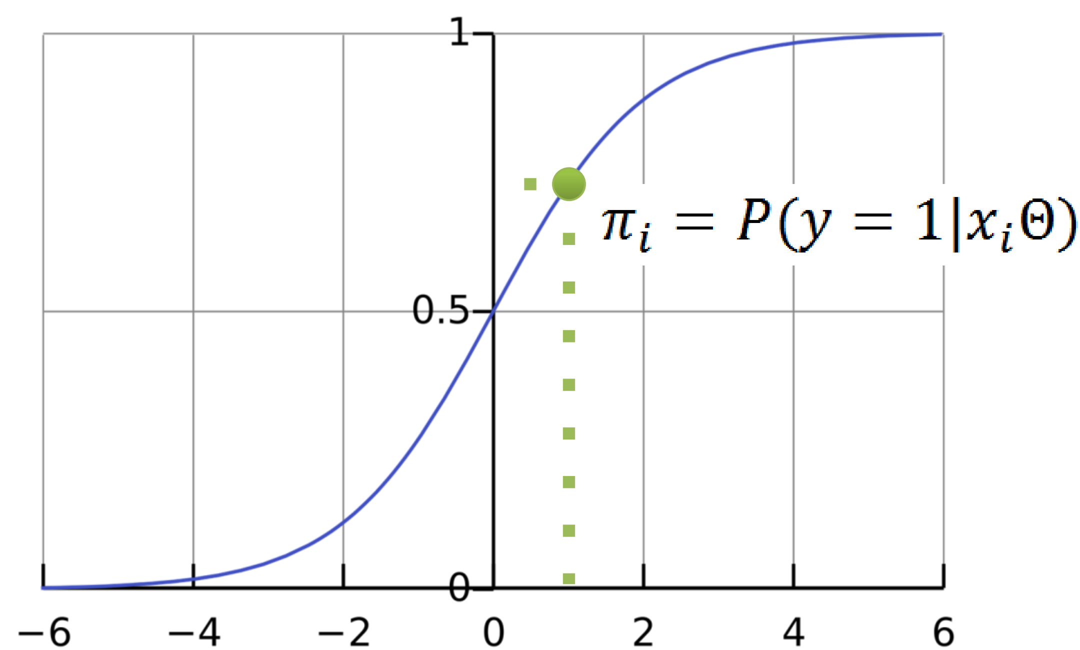
Sigmoid neuron



$$\frac{1}{1+e^{-X\theta}}$$

$$\pi_i = sigm(\eta) = \frac{1}{1+e^{-x\theta}}$$

$$y = \{0{,}1\}$$

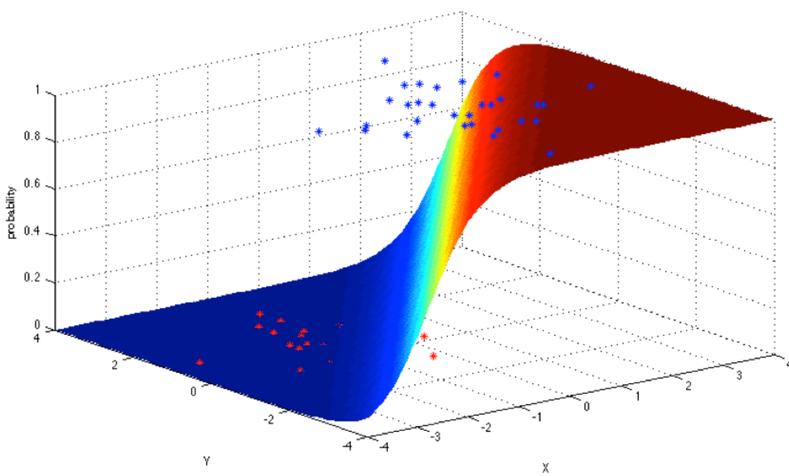


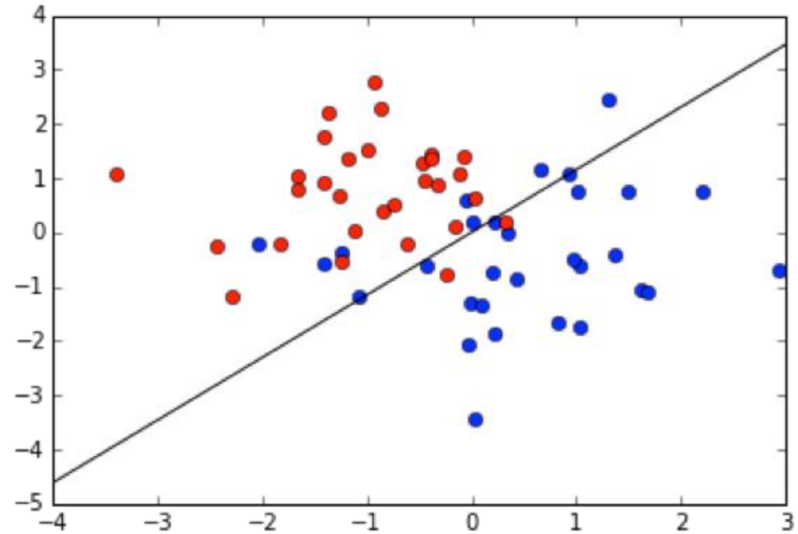
Linear separating hyper-plane

$$P(y = 1|X_i, \Theta) = \text{sigm}(X_i\Theta)$$

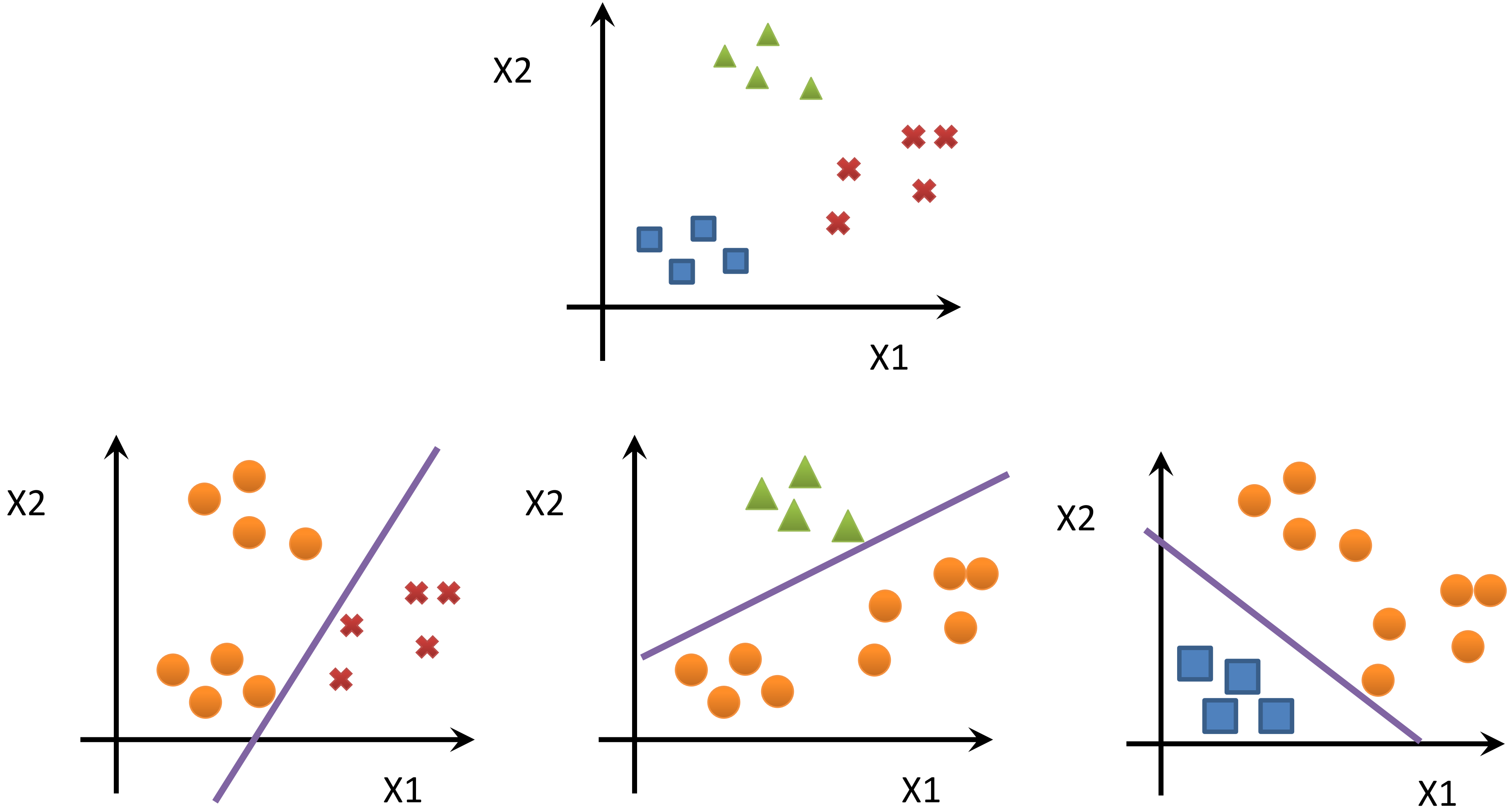
$$\text{sigm}(X_i\Theta) = \frac{1}{2} \quad \Leftrightarrow \quad X_i\Theta = 0 \qquad \frac{1}{1 + e^{-X\theta}}$$

x1	x2	y
-2.0	4.0	0
2.0	2.6	1
-0.4	1.0	1
…	…	….





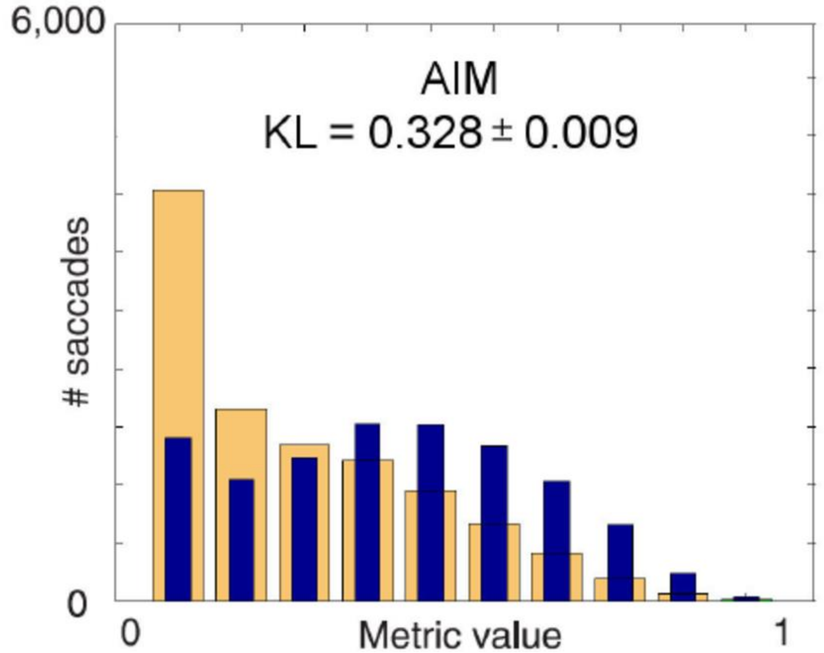
1 vs All



Error for the multiclass form

- C(θ) = f(y-y`, θ) $= -\,log\left(h(x)\right)^{y} - log\left(1 - h(x)\right)^{1-y}$

- C(θ) = f(KL, θ)



Bernuli

- A Bernoulli random variable (r.v.) y takes values in {0,1}

$$p(y;\Pi) = \begin{cases} \Pi & y = 1 \\ 1 - \Pi & y = 0 \end{cases}$$

$$= \Pi^y (1 - \Pi)^{1-y}$$

- The logistic regression model species the probability of a binary output given the input xi as follows:

$$p(y|X;\theta) = \prod_{i=1}^{n} Ber(yi \,|sigm(x_i\theta))$$

$$= \prod_{i=1}^{n} \left[\underbrace{\frac{1}{1+e^{-x_i\theta}}}_{\Pi_i}\right]^{y_i} \left[\underbrace{1-\frac{1}{1+e^{-x_i\theta}}}_{1-\Pi_i}\right]^{1-y_i}$$

actual	1	0	1	0	0	0	1	0	0
Pred y=1	0.8	0.2	0.9	0.01	0.2	0.4	0.95	0.3	0.18

likelihood	0.8	0.8	0.9	0.99	0.8	0.6	0.95	0.7	0.82

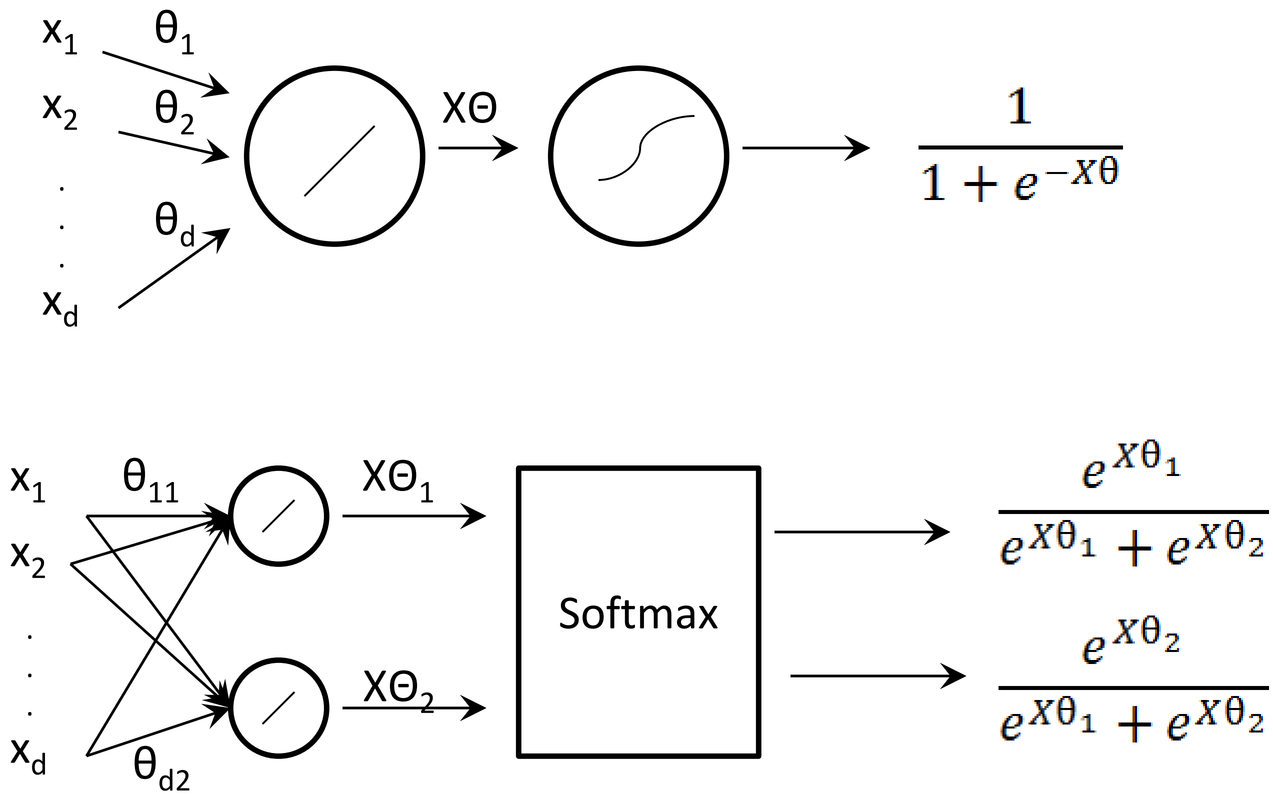
Maximum Likelihood Estimation

$$c(\theta) = -\log P(y|x,\theta)$$

$$= -\log \prod_{i=1}^{n} \left[\frac{1}{1+e^{-x_i\theta}}\right]^{y_i} \left[1 - \frac{1}{1+e^{-x_i\theta}}\right]^{1-y_i}$$

$$= -\sum_{i=1}^{n} y_i \log \Pi_i + (1-y_i)\log(1-\Pi_i)$$

Softmax



x_1
θ_1
x_2
θ_2
θ_d
x_d
$X\Theta$
$\frac{1}{1+e^{-X\theta}}$
x_1
θ_{11}
x_2
x_d
θ_{d2}
$X\Theta_1$
$X\Theta_2$
Softmax
$\frac{e^{X\theta_1}}{e^{X\theta_1}+e^{X\theta_2}}$
$\frac{e^{X\theta_2}}{e^{X\theta_1}+e^{X\theta_2}}$

Likelihood function

Indicator:
$$\mathbb{I}_c\left(y^{(i)}\right) = \begin{cases} 1 & y^{(i)} = c \\ 0 & otherwise \end{cases}$$

$$p(y; x, \theta) = \prod_{i=0}^{n} \Pi_1^{(i)\mathbb{I}_0(y^{(i)})} \Pi_2^{(i)\mathbb{I}_1(y^{(i)})}$$

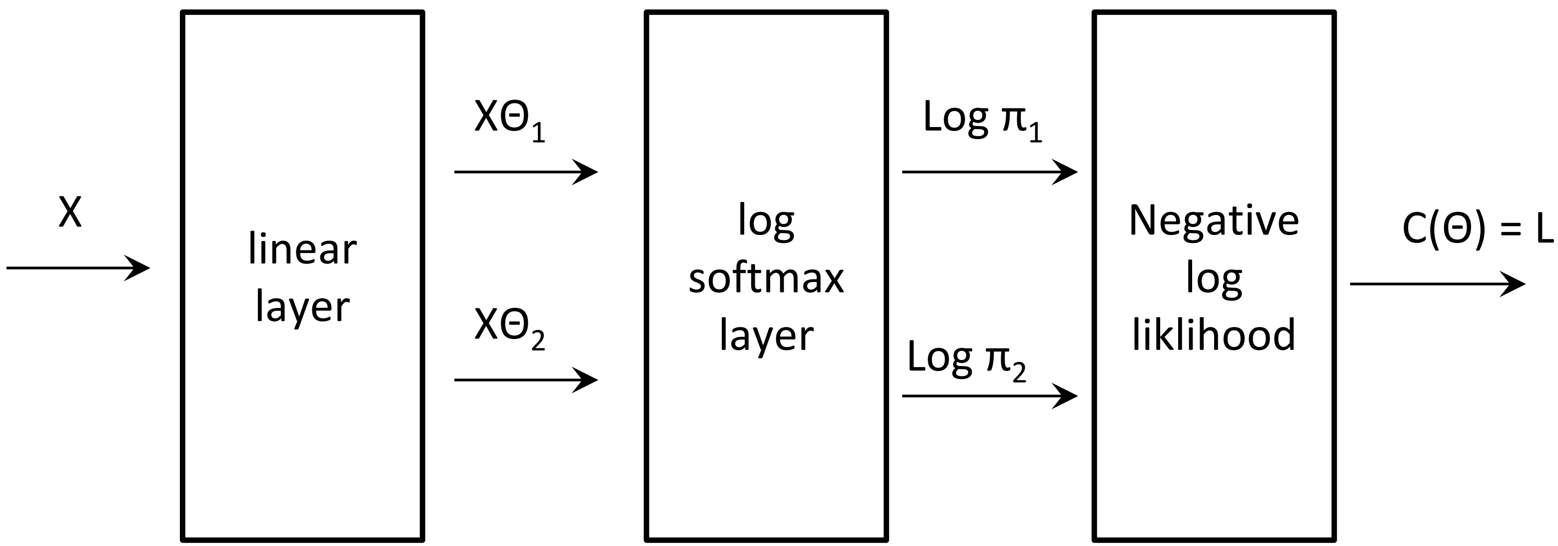
$$\mathrm{p}\left(y^{(i)} | x^{(i)}, \theta\right) = \begin{cases} \Pi_1^{(i)} = \dfrac{e^{x^{(i)}\theta_1}}{e^{x^{(i)}\theta_1} + e^{x^{(i)}\theta_2}} & if\ y = 1 \\ \Pi_2^{(i)} = \dfrac{e^{x^{(i)}\theta_2}}{e^{x^{(i)}\theta_1} + e^{x^{(i)}\theta_2}} & if\ y = 2 \end{cases}$$

$$p(y;x,\theta) = \prod_{i=0}^{n} \Pi_1^{(i)\mathbb{I}_0(y^{(i)})} \Pi_2^{(i)\mathbb{I}_1(y^{(i)})}$$

$$\mathbb{I}_c\left(y^{(i)}|x^{(i)},\theta\right) = \begin{cases} \Pi_1^{(i)} = \dfrac{e^{x^{(i)}\theta_1}}{e^{x^{(i)}\theta_1} + e^{x^{(i)}\theta_2}} & if\ y = 1 \\ \Pi_2^{(i)} = \dfrac{e^{x^{(i)}\theta_2}}{e^{x^{(i)}\theta_1} + e^{x^{(i)}\theta_2}} & if\ y = 2 \end{cases}$$

$$c(\theta) = -\log\bigl(p(y;x,\theta)\bigr) = -\sum_{i=1}^{n} \mathbb{I}_0\left(y^{(i)}\right) log\mathit{\Pi}_0^{(i)} + \mathbb{I}_1\left(y^{(i)}\right) log\mathit{\Pi}_1^{(i)}$$

Neural network representation of loss



Chain Rule – Quick reminder

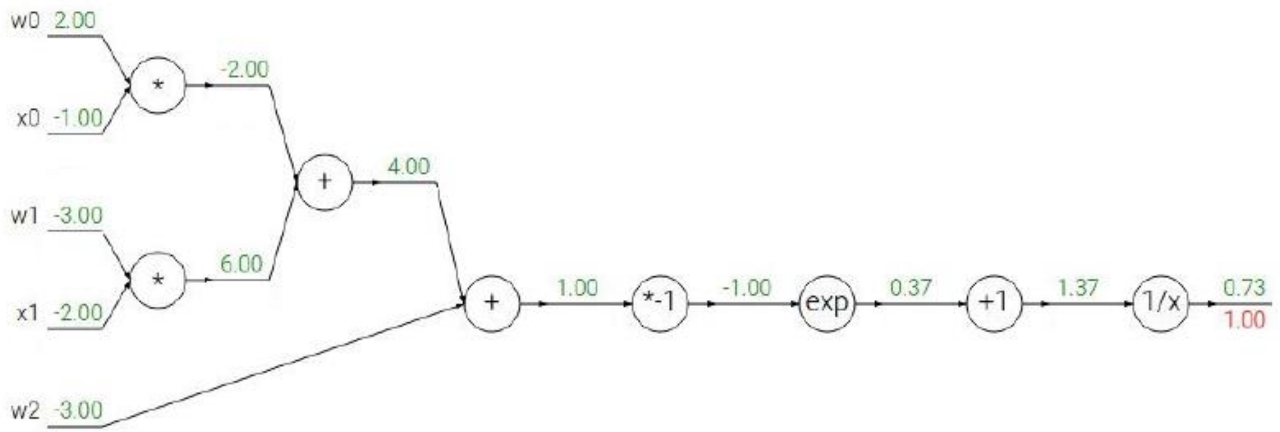
Compound expressions: $f(x,y,z) = (x+y)z$

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

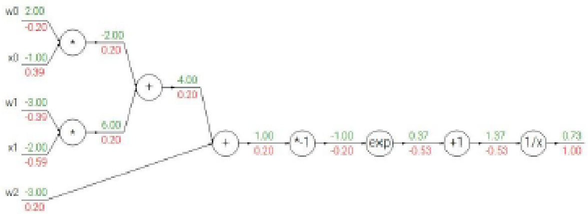


$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x \qquad\qquad f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a \qquad\qquad f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

תרגיל

Implementation: forward/backward API

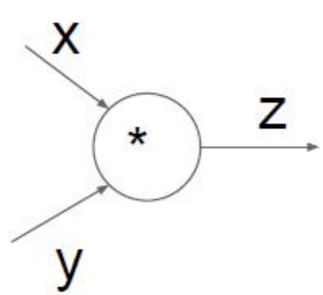


Graph (or Net) object. *(Rough psuedo code)*

```
class ComputationalGraph(object):
    #...
    def forward(inputs):
        # 1. [pass inputs to input gates...]
        # 2. forward the computational graph:
        for gate in self.graph.nodes_topologically_sorted():
            gate.forward()
        return loss # the final gate in the graph outputs the loss
    def backward():
        for gate in reversed(self.graph.nodes_topologically_sorted()):
            gate.backward() # little piece of backprop (chain rule applied)
        return inputs_gradients
```

תרגיל

Implementation: forward/backward API



(x,y,z are scalars)

```
class MultiplyGate(object):
    def forward(x,y):
        z = x*y
        return z
    def backward(dz):
        # dx = ... #todo
        # dy = ... #todo
        return [dx, dy]
```

$\boxed{\frac{\partial L}{\partial z}}$

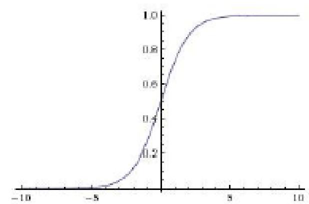
$\boxed{\frac{\partial L}{\partial x}}$

More activation functions

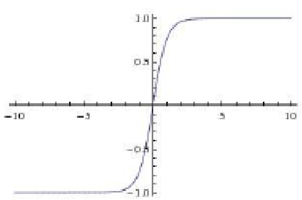
Activation Functions

Sigmoid

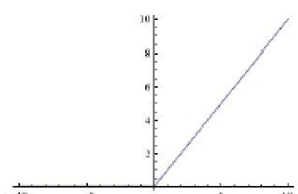
$$\sigma(x) = 1/(1+e^{-x})$$



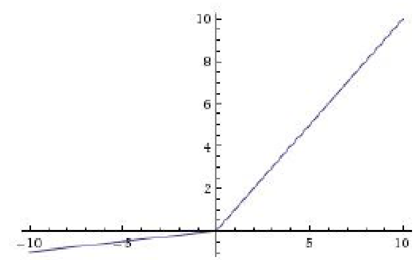
tanh tanh(x)



ReLU max(0,x)



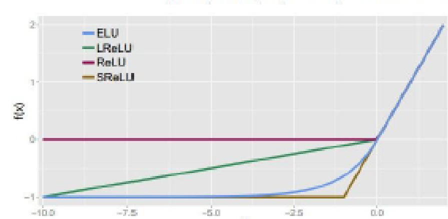
Leaky ReLU
max(0.1x, x)



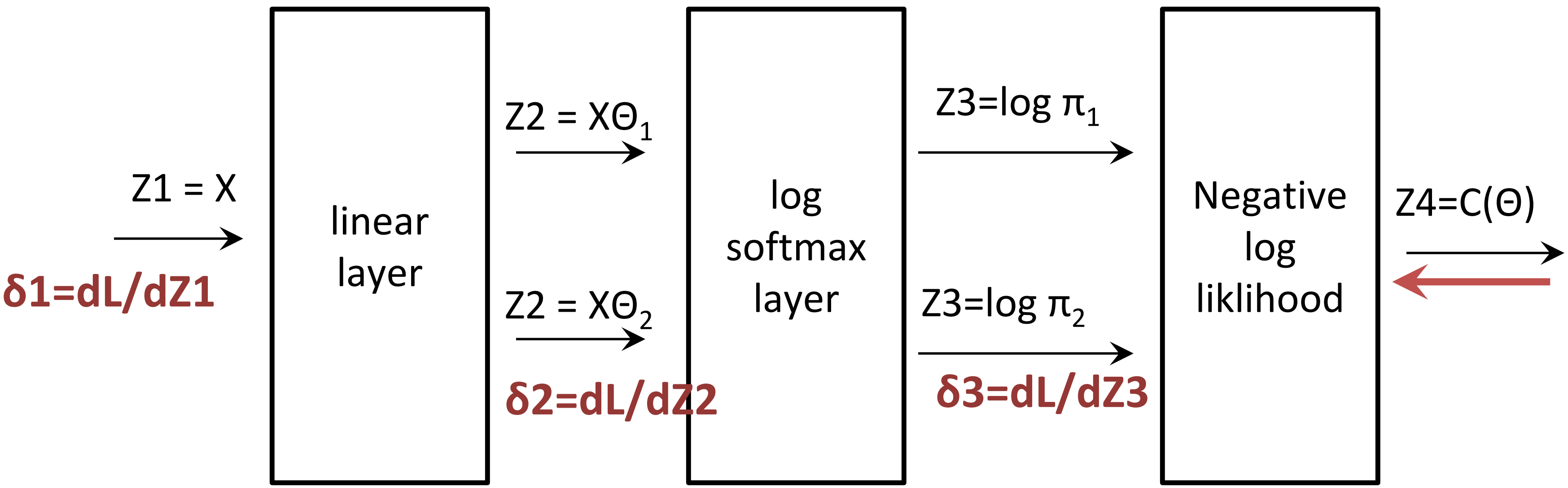
Maxout $\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \le 0 \end{cases}$$



Layer-wise design

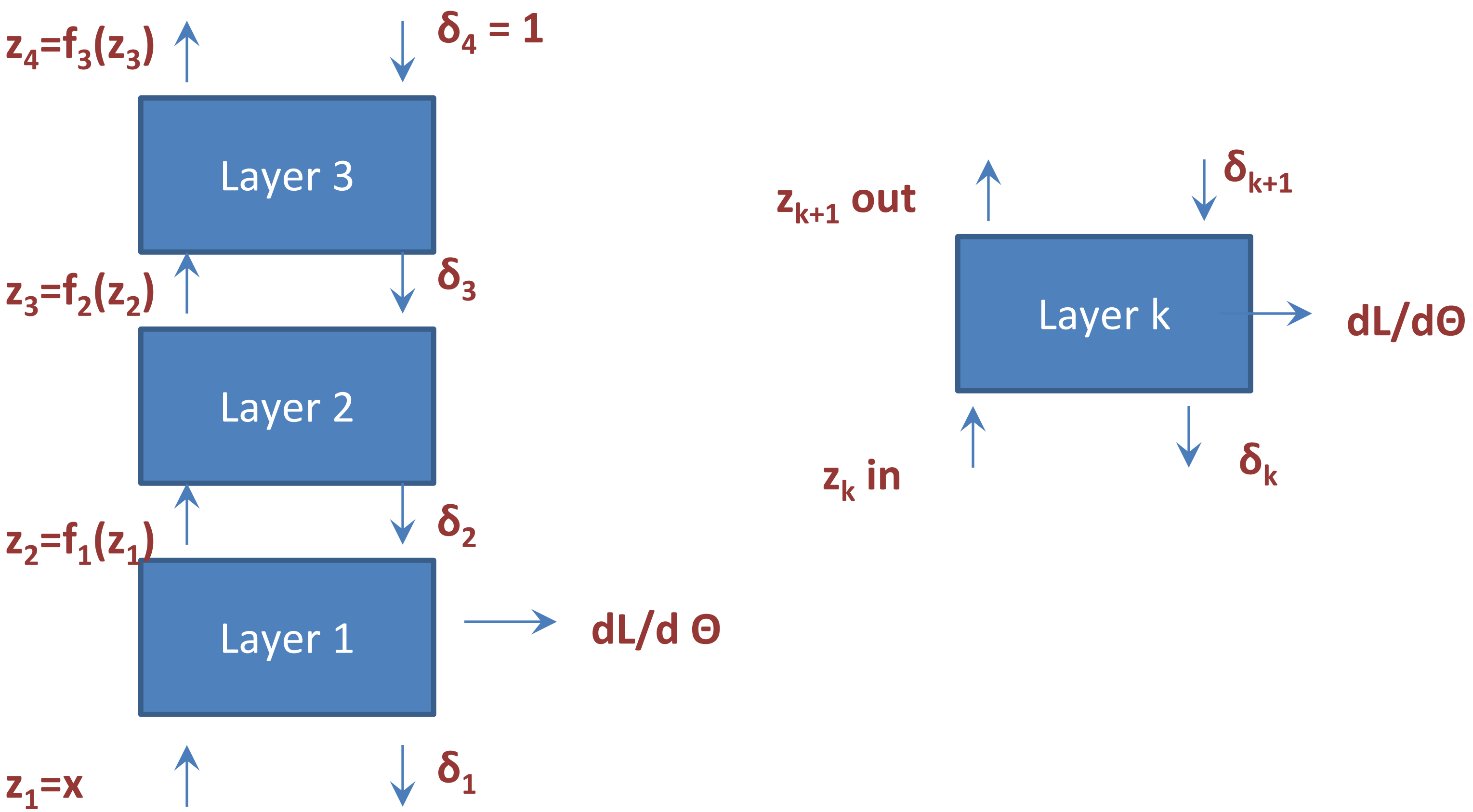


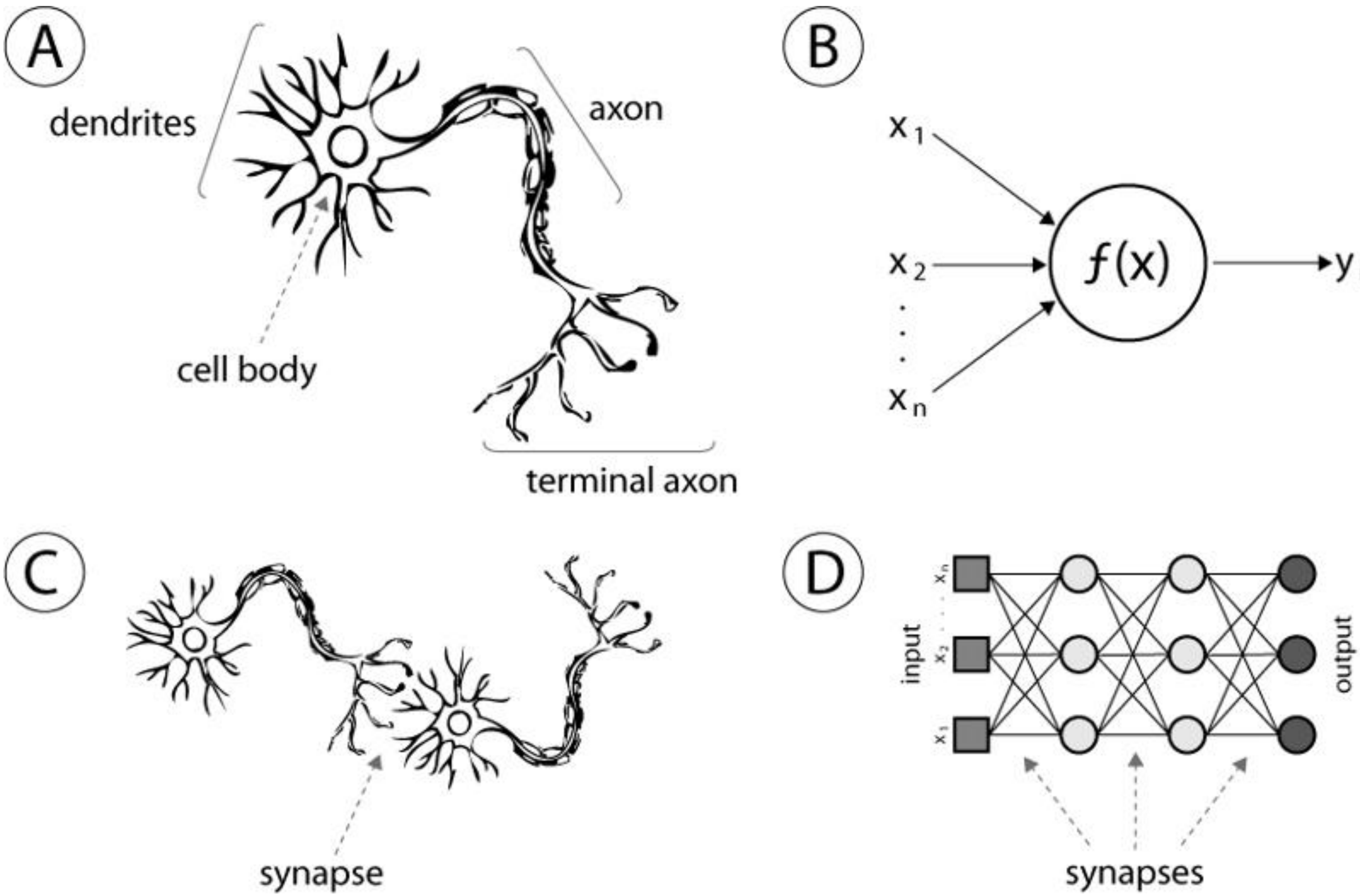
$$z_1^2 = x^{(i)}\theta_1 \qquad z_2^2 = x^{(i)}\theta_2$$

$$z_1^3 = \frac{e^{z_1^2}}{e^{z_1^2} + e^{z_2^2}} \qquad z_2^3 = \frac{2}{e^{z_1^2} + e^{z_2^2}}$$

$$z^4 = \sum_i \mathbb{I}_1\left(y^{(i)}\right) z_1^3 + \mathbb{I}_2\left(y^{(i)}\right) z_2^3$$

Layer-wise design




A
dendrites
axon
cell body
terminal axon
B
x 1
x 2
x n
f(x)
y i
C
synapse
D
input
x n
x 2
x 1
output
synapses

בחיים האמיתיים

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.optimizers import SGD
model = Sequential()
# Dense(64) is a fully-connected layer with 64 hidden units.
model.add(Dense(64, input_dim=20, init='uniform'))
model.add(Activation('tanh'))
model.add(Dense(64, init='uniform'))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(10, init='uniform'))
model.add(Activation('softmax'))

sgd = SGD(lr=0.1, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd)
model.fit(X_train, y_train, nb_epoch=20, batch_size=16, show_accuracy=True)
score = model.evaluate(X_test, y_test, batch_size=16)
```