Deep Learning

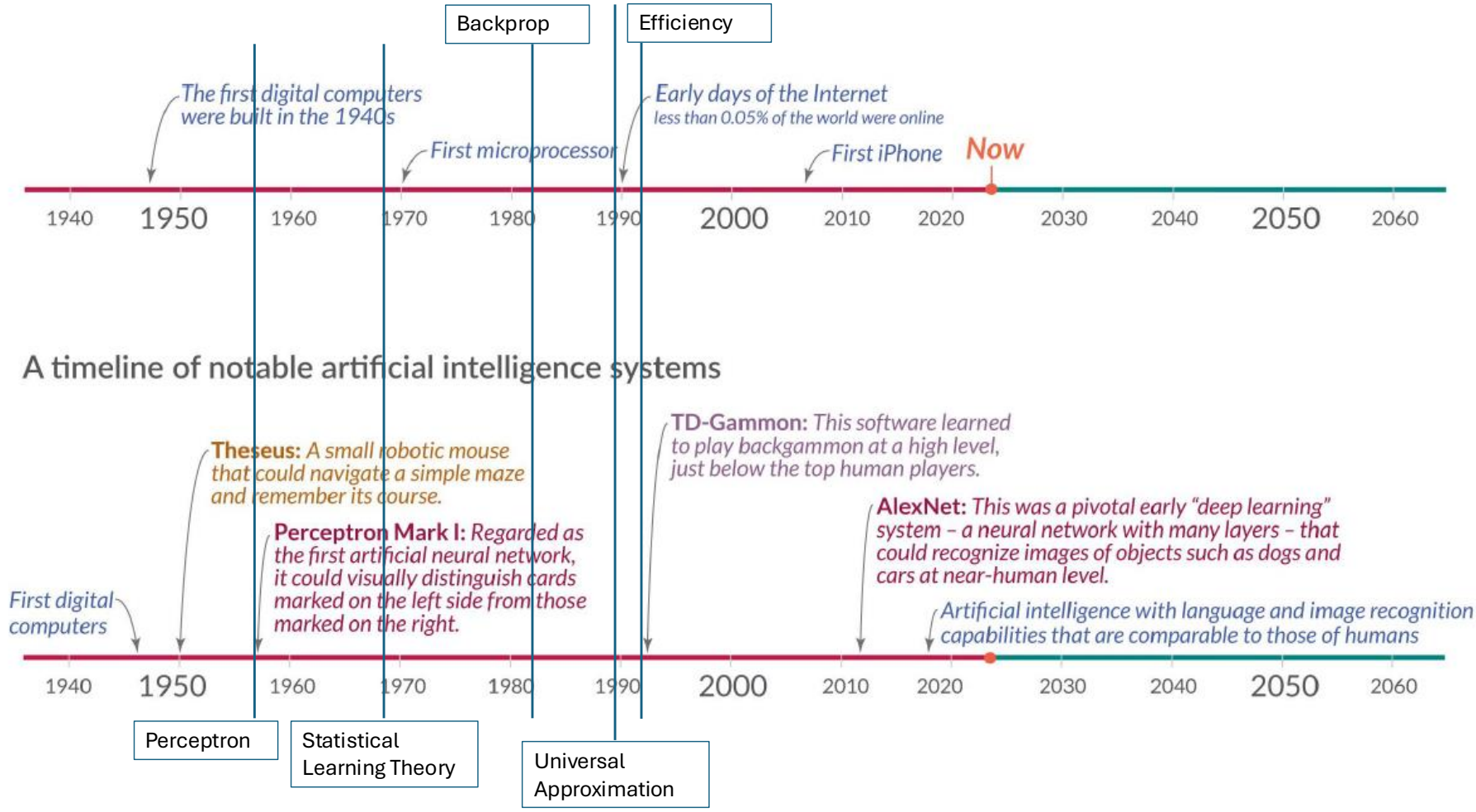
Topic 1 - History of Deep Learning, Deep Neural Networks, Backpropagation

Ivan Tyukin, DrSc
i.tyukin@skoltech.ru

Term 4, 2024-2025 AY

Course structure

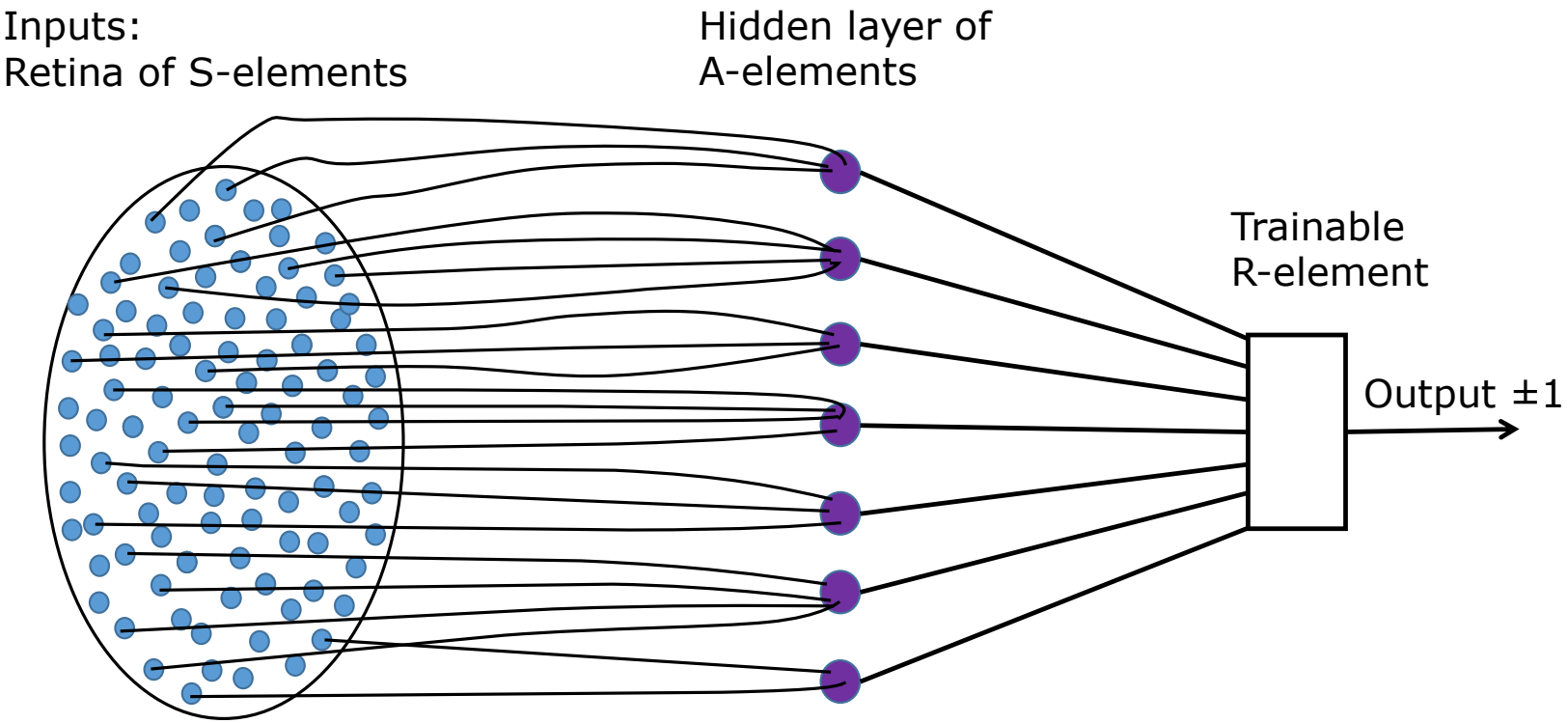
- **Topic 1: Introduction**
- Topic 2: Convolutional neural nets
- Topic 3: Computer vision with Deep Learning
- Topic 4: Recurrent Nets and Transformers
- Topic 5: Computer vision with Transformers
- Topic 6: Check-pointing, low-precision training
- Topic 7: Contrastive learning
- Topic 8: Stability and Adversarial Attacks
- Topic 9: Auto-encoders and GANs


Backprop
Efficiency
The first digital computers were built in the 1940s
First microprocessor
Early days of the Internet
less than 0.05% of the world were online
First iPhone
Now
1940
1950
1960
1970
1980
1990
2000
2010
2020
2030
2040
2050
2060
A timeline of notable artificial intelligence systems
Theseus: A small robotic mouse that could navigate a simple maze and remember its course.
Perceptron Mark I: Regarded as the first artificial neural network, it could visually distinguish cards marked on the left side from those marked on the right.
TD-Gammon: This software learned to play backgammon at a high level, just below the top human players.
AlexNet: This was a pivotal early "deep learning" system – a neural network with many layers – that could recognize images of objects such as dogs and cars at near-human level.
First digital computers
Artificial intelligence with language and image recognition capabilities that are comparable to those of humans
Perceptron
Statistical Learning Theory
Universal Approximation

Perceptron

- The perceptron was invented in 1943 by Warren McCulloch and Walter Pitts.
- In 1949, Donald Hebb postulated the first rule for self-organized learning.
- The first implementation was a machine built in 1958 at the Cornell Aeronautical Laboratory by Frank Rosenblatt, funded by the United States Office of Naval Research.
- Rosenblatt formulated a learning rule for the perceptron, related to Hebb's rule.
- In 1969 Minsky and Papert emphasised geometric aspects of perceptron learning.

Perceptron


Inputs:
Retina of S-elements
Hidden layer of
A-elements
Trainable
R-element
Output ±1

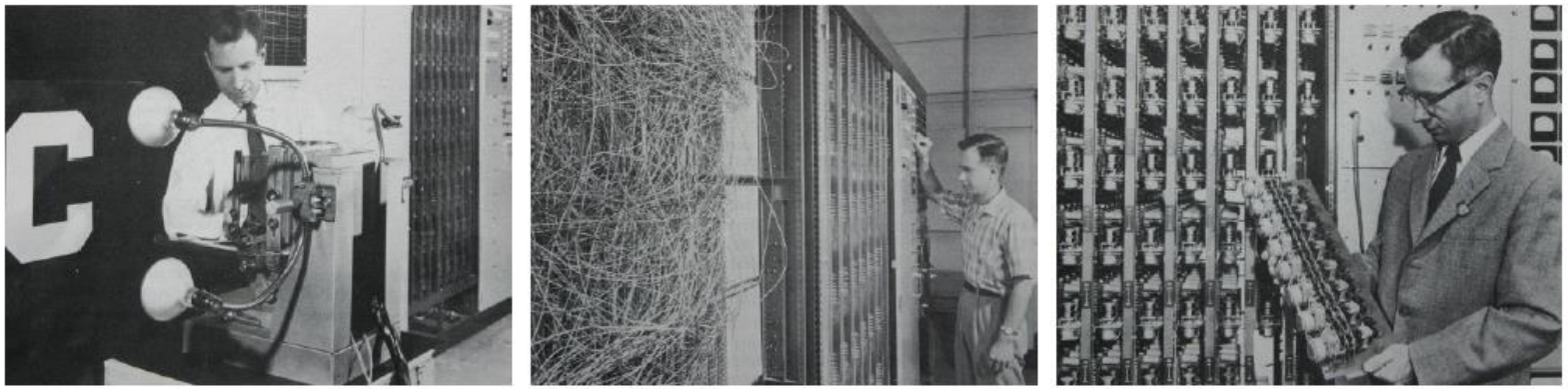
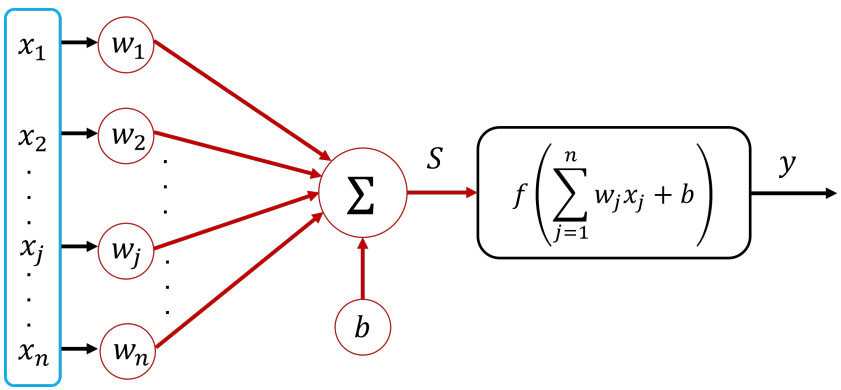


Figure 1.14 Illustration of the Mark 1 perceptron hardware. The photograph on the left shows how the inputs were obtained using a simple camera system in which an input scene, in this case a printed character, was illuminated by powerful lights, and an image focused onto a 20×20 array of cadmium sulphide photocells, giving a primitive 400-pixel image. The perceptron also had a patch board, shown in the middle photograph, which allowed different configurations of input features to be tried. Often these were wired up at random to demonstrate the ability of the perceptron to learn without the need for precise wiring, in contrast to a modern digital computer. The photograph on the right shows one of the racks of learnable weights. Each weight was implemented using a rotary variable resistor, also called a potentiometer, driven by an electric motor thereby allowing the value of the weight to be adjusted automatically by the learning algorithm.

(Taken from C. Bishop, H. Bishop. Deep Learning. Foundations and Concepts. Springer, 2024)

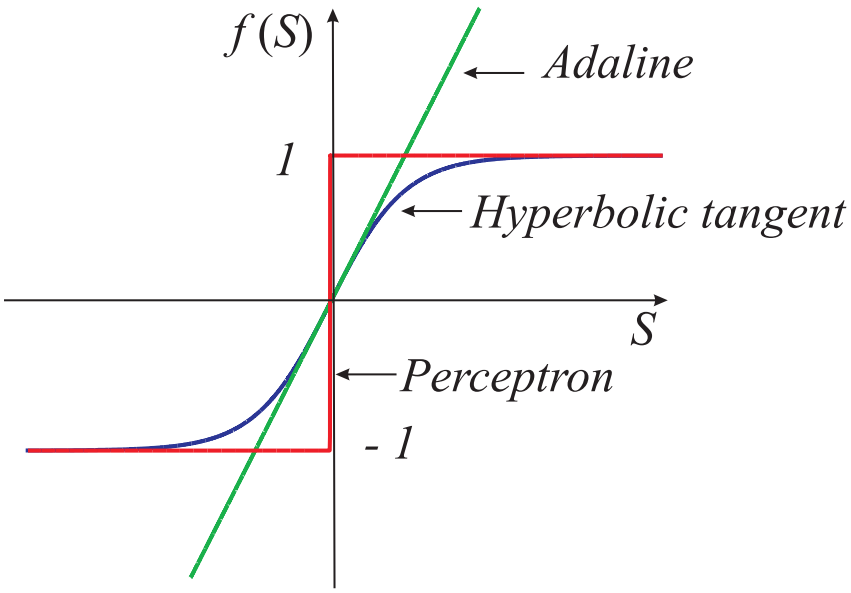
Perceptron

Schematic representation of the perceptron unit

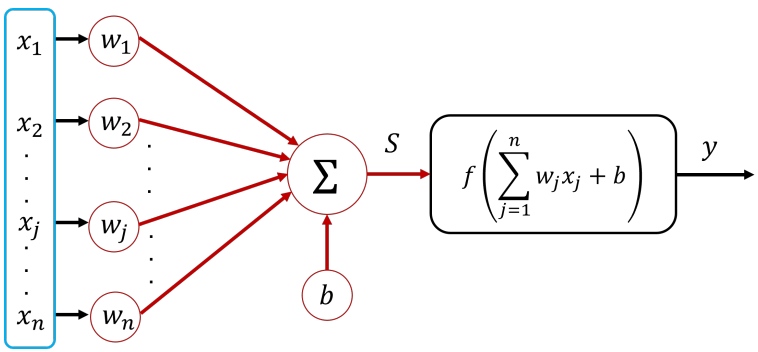


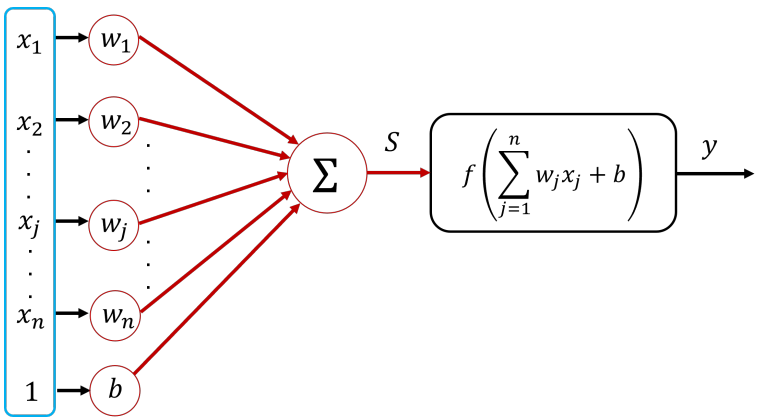
Activation functions

We will distinguish between three types of the artificial neurons depending on the form of the activation function:



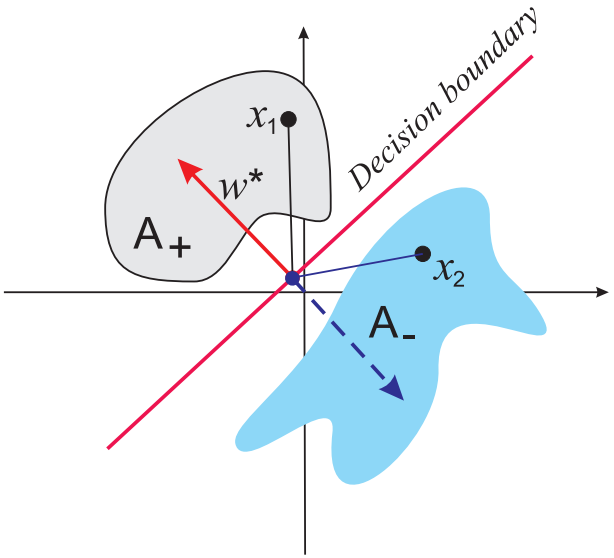
Decision variable





Perceptron

The goal of the perceptron is correctly classify the set of externally applied stimuli $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$ into one of two classes, C_1 or C_2.



Let A_+ and A_- be two disjoint subsets of $\mathbf{R}^n$

$$A_+ \cap A_- = \varnothing, \; A_+, A_- \subset \mathbf{R}^n$$

Let A_+ and A_- be linearly separable, i.e.

$\exists\, \mathbf{w}^* \in W^* \subset \mathbf{R}^n$:

$$\mathbf{w}^{*T}\mathbf{x} > 0 \,\forall\, \mathbf{x} \in A_+ \quad \text{and} \quad \mathbf{w}^{*T}\mathbf{x} < 0 \;\; \forall\, \mathbf{x} \in A_-$$

Then,

$$y = f(S) = \text{sgn}(\mathbf{w}^{*T}\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in A_+ \\ -1 & \mathbf{x} \in A_- \end{cases} \tag{1}$$

Perceptron learning algorithm

Let $\mathcal{W}^*$ be a set of all $\mathbf{w}^* \in \mathbf{R}^n$ such that ${\mathbf{w}^*}^T\mathbf{x} = 0$ separate A_+ and A_- (condition (1) holds). Suppose that:

1) $\{\mathbf{x}_i\}_{i=1}^N$, $N > 1$, $N \in \mathbb{N}$ be a sequence of points:

$$\mathbf{x}_i \in A_+ \quad \text{or} \quad \mathbf{x}_i \in A_- \;\forall\, i \in \{1, \ldots, N\}.$$

2) $\{y_i\}_{i=1}^N$ be a sequence of "true responses" for $\{\mathbf{x}_i\}_{i=1}^N$:

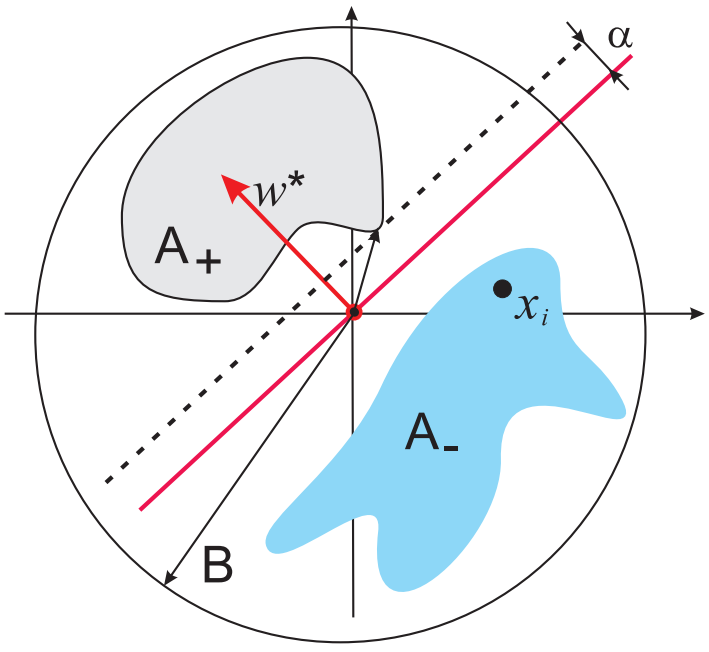
$$y_i = 1 \Leftrightarrow \mathbf{x}_i \in A_+; \quad y_i = -1 \Leftrightarrow \mathbf{x}_i \in A_-.$$

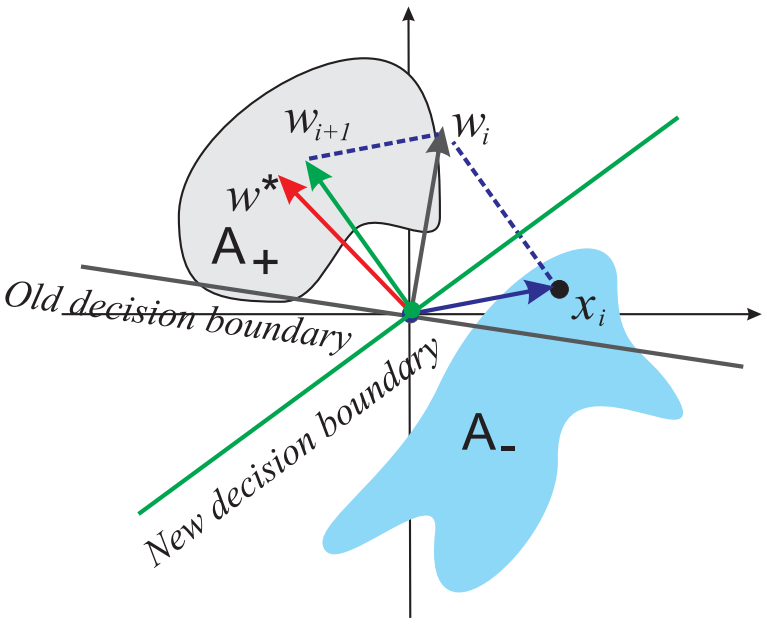
Question

Is there a function $\mathcal{A}(\mathbf{x}_i, y_i, \mathbf{w}_i)$*:*

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \mathcal{A}(\mathbf{x}_i, y_i, \mathbf{w}_i)$$

such that for all $i \geq k_{max}$, $1 \leq k_{max} < \infty$, $\mathbf{w}_i \in \mathcal{W}^*$ *?*


α
w*
A+
A-
x_i
B


w_{i+1}
w_i
w*
A+
Old decision boundary
x_i
New decision boundary
A-

Perceptron learning algorithm

Algorithm:

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \begin{cases} 0 & \text{sgn}(\mathbf{x}_i^T \mathbf{w}_i) y_i > 0 \\ y_i \mathbf{x}_i & \text{sgn}(\mathbf{x}_i^T \mathbf{w}_i) y_i \leq 0 \end{cases}, \quad \mathbf{w}_0 = 0$$

$\mathbf{w}_0 = 0$ - the initial state of the weights.

Example

Problem

Consider the following data:

x_1	x_2	*Class*	x_1	x_2	*Class*
1	*2*	*1*	*3*	*1*	*-1*
3	*3*	*1*	*4*	*2*	*-1*

Implement several steps of the perceptron training algorithm.

Theorem (Novikoff, A.B. 1962)

Let A_+, A_- be given and $\mathcal{W}^* \neq \varnothing$. Suppose that

$$\exists\ B > 0,\ B \in \mathbf{R}:\ \|\mathbf{x}\| \leq B \quad \forall\ \mathbf{x} \in A_+ \cup A_-$$

and for some $\mathbf{w}^* \in \mathcal{W}^*$ the following holds:

$$\min_{\mathbf{x} \in A_+ \cup A_-} \{|{\mathbf{w}^*}^T \mathbf{x}|\} = \alpha.$$

Then elements of sequence

$$\mathbf{w}_k = \mathbf{w}_{k-1} + \begin{cases} 0 & \operatorname{sgn}(\mathbf{w}_{k-1}^T \mathbf{x}_k) y_k > 0 \\ y_k \mathbf{x}_k & \operatorname{sgn}(\mathbf{w}_{k-1}^T \mathbf{x}_k) y_k \leq 0 \end{cases}, \quad \mathbf{w}_0 = 0 \tag{2}$$

converge into $\mathcal{W}^*$ after at most $k_{max} \leq [B^2 \|\mathbf{w}^*\|^2 / \alpha^2] + 1$ corrections.

Proof

Step 1 Notice that

$$y_i \mathbf{w}^{*T} \mathbf{x}_i = \text{sgn}(\mathbf{w}^{*T} \mathbf{x}_i) \mathbf{w}^{*T} \mathbf{x}_i \geq \alpha > 0$$

for any $\mathbf{x}_i$ in the training set

Proof

Step 2. Without the loss of generality, assume that the first k iterations require correction, i.e. $\text{sgn}(\mathbf{w}_{i-1}^T \mathbf{x}_i) y_i < 0$ for $i = 1, 2, ..., k$ (otherwise we can "skip" iterations without corrections and redefine indices of $\mathbf{x}_i$ accordingly)

Consider ${\mathbf{w}^*}^T \mathbf{w}_k$ for the k-th correction, where

$$ {\mathbf{w}^*}^T \mathbf{w}_k = {\mathbf{w}^*}^T (\mathbf{w}_{k-1} + y_k \mathbf{x}_k) = {\mathbf{w}^*}^T \mathbf{w}_{k-1} + y_k {\mathbf{w}^*}^T \mathbf{x}_k \geq {\mathbf{w}^*}^T \mathbf{w}_{k-1} + \alpha $$

Applying the same argument $k - 1$ times:

$$ {\mathbf{w}^*}^T \mathbf{w}_k \geq {\mathbf{w}^*}^T \mathbf{w}_{k-2} + 2\alpha $$

$$ \dots $$

and taking into account that $\mathbf{w_0} = 0$, we derive that:

$$ {\mathbf{w}^*}^T \mathbf{w}_k \geq k\alpha. $$

Proof

Step 3: Let us compute $\|\mathbf{w}_{k+1}\|^2$:

$$\begin{aligned}\|\mathbf{w}_k\|^2 &= \|\mathbf{w}_{k-1} + y_k\mathbf{x}_k\|^2 = \|\mathbf{w}_{k-1}\|^2 + 2y_k\mathbf{w}_{k-1}^T\mathbf{x}_k + \|\mathbf{x}_k\|^2 \\ &< \|\mathbf{w}_{k-1}\|^2 + \|\mathbf{x}_k\|^2 \leq \|\mathbf{w}_{k-1}\|^2 + B^2 \leq kB^2\end{aligned}$$

So far we have:

$$\|\mathbf{w}_k\|^2 < kB^2, \quad \mathbf{w}^{*T}\mathbf{w}_k \geq k\alpha$$

Proof

Step 4:

$$1 \geq |\cos(\widehat{\mathbf{w}^*, \mathbf{w}_k})| = \frac{|\mathbf{w}^{*T}\mathbf{w}_k|}{\|\mathbf{w}^*\|\|\mathbf{w}_k\|} > \frac{k\alpha}{\sqrt{k}B\|\mathbf{w}^*\|}$$

Hence the number of corrections should always satisfy the following inequality

$$\sqrt{k} \leq \frac{B\|\mathbf{w}^*\|}{\alpha} \Rightarrow k \leq \frac{B^2\|\mathbf{w}^*\|^2}{\alpha^2}$$

Thus perceptron learning rule specified by (2) needs no more than

$$k_{max} = \left[\frac{B^2\|\mathbf{w}^*\|^2}{\alpha^2}\right] + 1$$

corrections to separate A_+ and A_-.

□

Remark

The perceptron convergence theorem states that if ***there exists an exact solution****, then the perceptron learning algorithm is guaranteed to find an exact solution in a finite number of steps.*

Remark

1. *Sufficiently reach data is needed: to use the rule we must ensure that at least*

$$k_{max} = \left[\frac{B^2 \|\mathbf{w}^*\|^2}{\alpha^2} \right] + 1$$

   corrections will be made.
2. *The estimate holds for* $\mathbf{w}_0 = 0$.

problems for revision

Problem

Consider the following data:

x_1	x_2	*Class*	x_1	x_2	*Class*
2	*1*	*1*	*-2*	*0*	*-1*
1	*2*	*1*	*-1*	*1*	*-1*
1	*1*	*1*	*0*	*-1*	*-1*

Implement several steps of the perceptron training algorithm.

Problem

- *Which logical operations (AND, OR, XOR) can the perceptron unit learn?*

What do we mean by "learning" ?

- Data sample (i.i.d.) from some unknown $P(y, \mathbf{x})$

$$(\mathbf{x}_1, y_1),\ (\mathbf{x}_2, y_1), \ldots (\mathbf{x}_m, y_m) \in \mathcal{D} \subset \mathbb{R}^n \times L$$

- Model: $f : \mathcal{D} \to L$
- Loss and Risk functionals

$$R(f) = \int \mathcal{L}(y, f(\mathbf{x}))dP(y, \mathbf{x})$$

Problems?

Right, we cannot compute risk without knowing $P(y, \mathbf{x})$, and hence the need for learning.

Instead of true risk $R(f)$, we can compute empirical risk

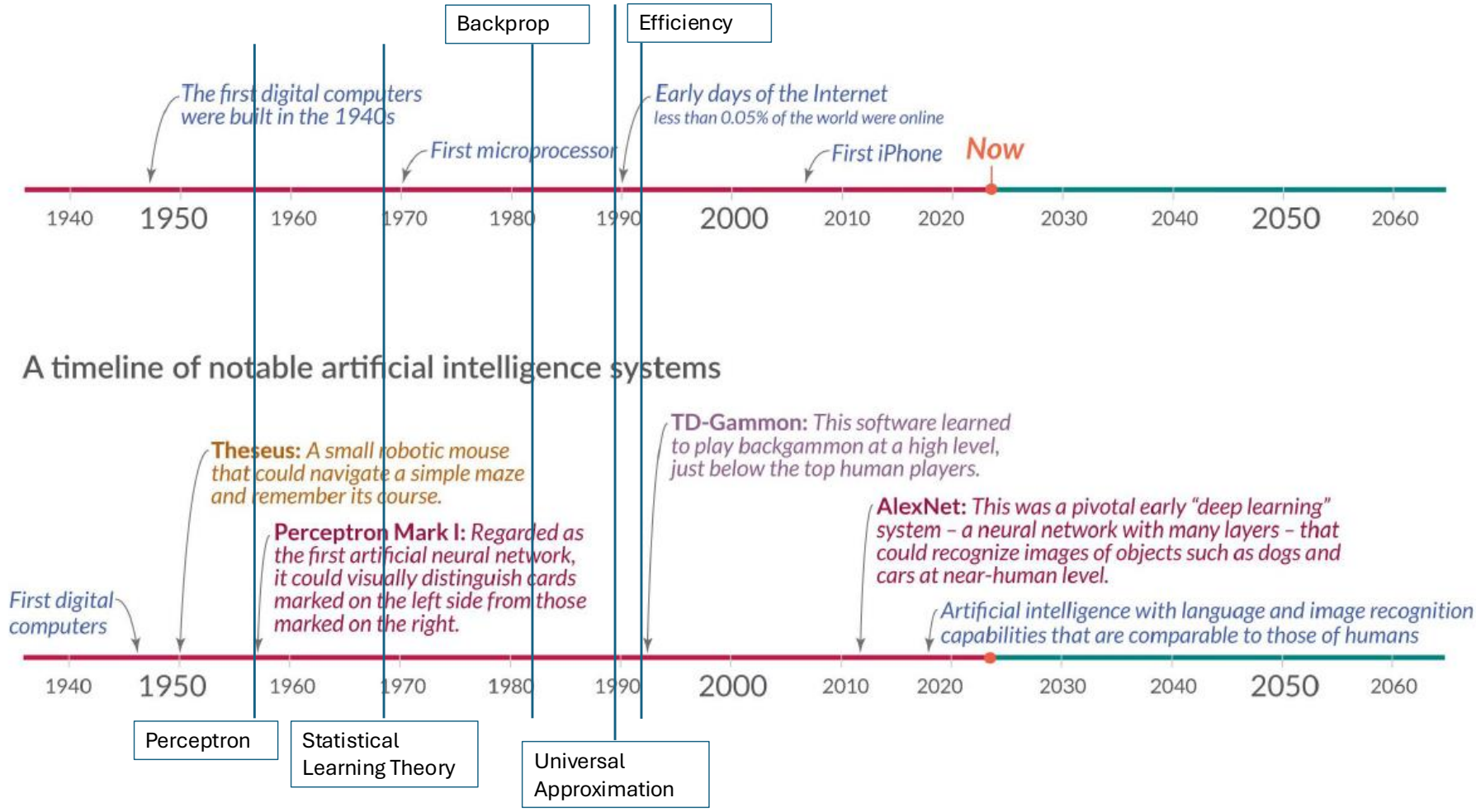
$$\hat{R}(f) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(y_i, f(\mathbf{x}_i))$$

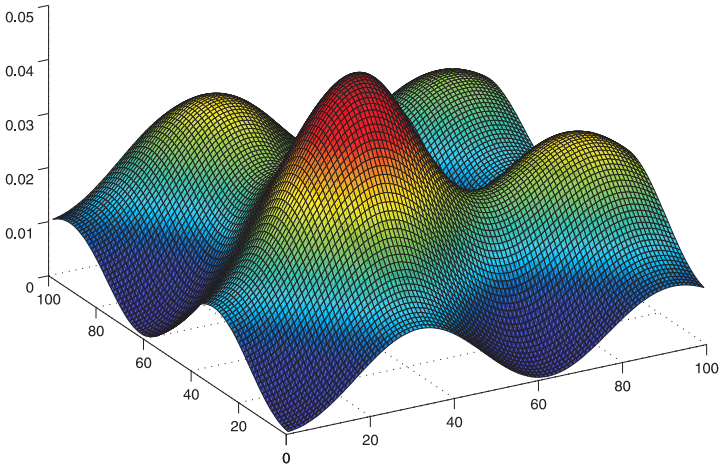
Statistical learning theory (Bousquet, O. et. al (2004). Introduction to Statistical Learning Theory. Lecture Notes in Computer Science, vol 3176. Springer, Berlin, Heidelberg) tells us, that for m sufficiently large, with probability $1 - \delta$ the following holds true:

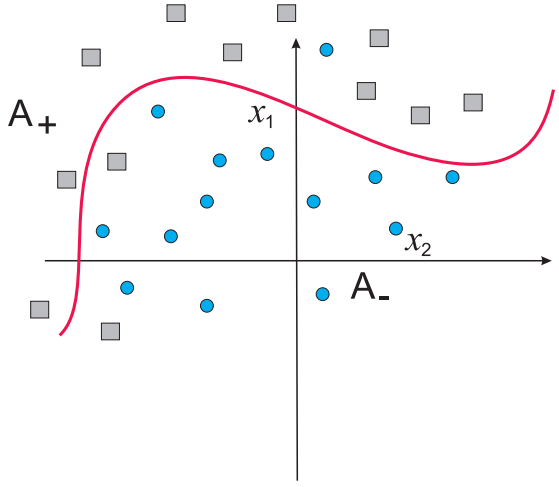
$$R(f) \leq \hat{R}(f) + 2\sqrt{\frac{2}{m}\left(h \ln(em/h) + \ln(4/\delta)\right)}$$

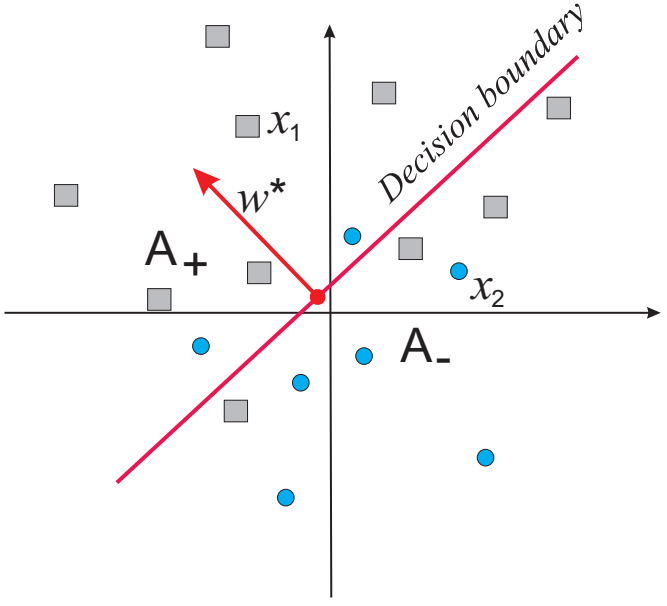
where h is the VC-dimension of the model f.

That is, minimizing $\hat{R}(f)$ on sufficiently large datasets amounts to learning (if h is finite).


Backprop
Efficiency
The first digital computers were built in the 1940s
First microprocessor
Early days of the Internet
less than 0.05% of the world were online
First iPhone
Now
1940
1950
1960
1970
1980
1990
2000
2010
2020
2030
2040
2050
2060
A timeline of notable artificial intelligence systems
Theseus: A small robotic mouse that could navigate a simple maze and remember its course.
Perceptron Mark I: Regarded as the first artificial neural network, it could visually distinguish cards marked on the left side from those marked on the right.
TD-Gammon: This software learned to play backgammon at a high level, just below the top human players.
AlexNet: This was a pivotal early "deep learning" system – a neural network with many layers – that could recognize images of objects such as dogs and cars at near-human level.
First digital computers
Artificial intelligence with language and image recognition capabilities that are comparable to those of humans
Perceptron
Statistical Learning Theory
Universal Approximation


0.05
0.04
0.03
0.02
0.01
0
100
80
60
40
20
0
20
40
60
80
100


A+
x1
x2
A-


x1
w*
Decision boundary
A+
x2
A-

Consider $\mathbf{x} \in [a,b]^d$, $a, b \in \mathbb{R}$, and let

$$\bar{f}(\mathbf{x}) : \ [a,b]^d \to \mathbb{R}$$

be a "model" of our data (I assume, for now that we can access the data as many times as needed, and there is no noise).

To reconstruct (unknown yet existing) f we need something

- simple (e.g. a collection of simple units or functions $\varphi \in \mathcal{G}$);
- that can approximate functions of many variables;
- with a good rate of convergence

Why "linear" series expansions are not always good (in L_2)

$$\lim_{n\to\infty} \min_{c_i} \|\bar{f}(\mathbf{x}) - \sum_{i=1}^{n} c_i \varphi_i(\mathbf{x})\| = 0, \;\; \varphi_i(\mathbf{x}) \in \mathcal{G}$$

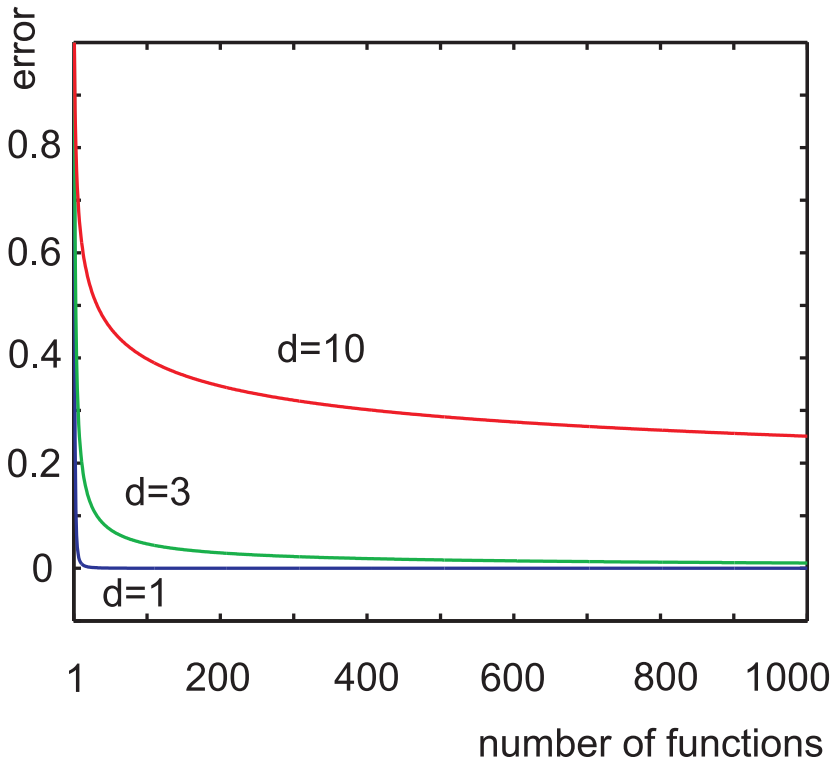
$$\|\bar{f}(\mathbf{x}) - \sum_{i=1}^{n} c_i \varphi_i(\mathbf{x})\|^2 \geq O\left(\frac{1}{n^{2/d}}\right)$$

Examples

- polynomials
- trigonometric functions

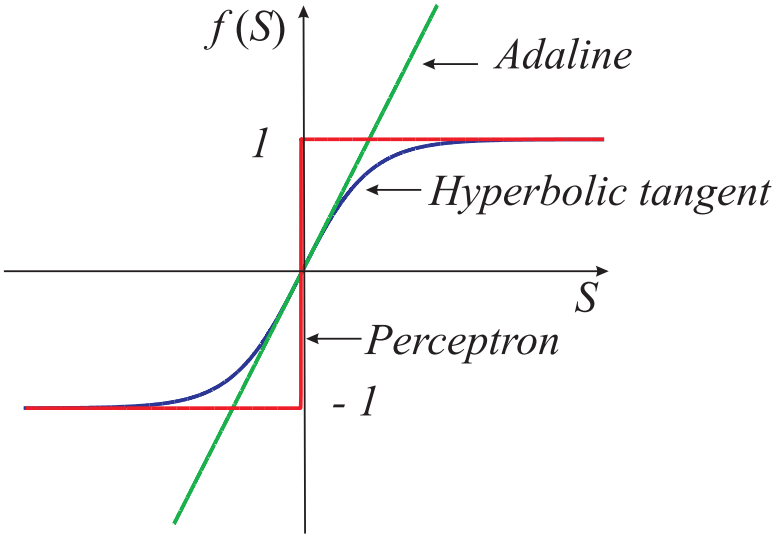
Approximation error decreases too slow for large d.

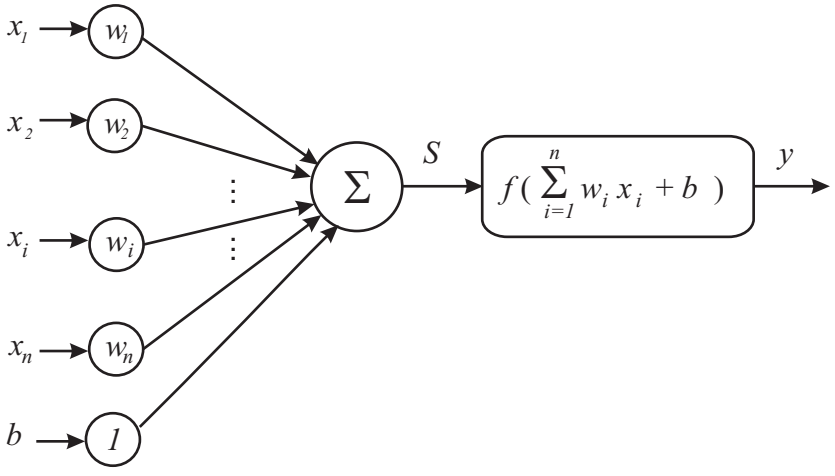
Convergence rates for $d = 1$, $d = 3$, $d = 10$



Approximating with "adaptive" as opposed to fixed set of functions

1. Approximation capabilities of sigmoid superpositions





Sigmoid-shaped functions

$$f(S) = \frac{1 - e^{-S}}{1 + e^{S}}, \quad f(S) = \frac{1}{1 + e^{-S}}, \quad f(S) = \arctan(S)$$

Just approximation first ...

Theorem (Cybenko, 1989)

Let $f(S)$ be a continuous sigmoid discriminatory function

$$f(S) \to \begin{cases} 1, & S \to \infty \\ 0, & S \to -\infty \end{cases}$$

and $\mathcal{C}[0,1]^d$ be a space of real-valued continuous functions defined on $[0,1]^d$. Then sums

$$\sum_{i=1}^{n} c_i f(\mathbf{w}_i^T \mathbf{x} + b_i), \ \mathbf{w}_i \in \mathbb{R}^d, \ c_i, b_i \in \mathbb{R}$$

are dense in $\mathcal{C}[0,1]^d$. That is for any $\varepsilon \in \mathbb{R}_{>0}$ and $\bar{f}(\mathbf{x}) \in \mathcal{C}[0,1]^d$ there exist $N \in \mathbb{N}$, $\mathbf{w}_i$, c_i, b_i:

$$\|\bar{f}(\mathbf{x}) - \sum_{i=1}^{N} c_i f(\mathbf{w}_i^T \mathbf{x} + b_i)\| \leq \varepsilon \ \forall \ \mathbf{x} \in [0,1]^d$$

We can easily prove this theorem for a special case when

- for $f(S)$: $\sigma_b(S) = f(S) - f(S+b)$ is Riemann-integrable

Step 1. Approximation in one dimension (sketch)

$$\bar{f}(v) = a_0 + \sum_{n=1}^{\infty} a_n \cos(nv) + b_n \sin(nv)$$

Consider

$$\hat{\sigma}_b(\omega) = \int_{-\infty}^{\infty} \sigma_b(S) e^{-i\omega S} dS$$

Pick ω': $\hat{\sigma}_b(\omega') \neq 0$ and consider $e^{iv}\hat{\sigma}_b(\omega') = e^{iv} \int_{-\infty}^{\infty} \sigma_b(S) e^{-i\omega' S} dS$

$$e^{iv}\hat{\sigma}_b(\omega') = e^{iv}\int_{-\infty}^{\infty}\sigma_b(S)e^{-i\omega' S}dS = \int_{-\infty}^{\infty}\sigma_b(S)e^{-i\omega' S}e^{iv}dS$$

$$e^{iv}\hat{\sigma}_b(\omega') = \int_{-\infty}^{\infty}\sigma_b(S)e^{-i(\omega' S - v)}dS$$

After changing the integration variable $x = \omega' S - v \Rightarrow dx = \omega' dS$:

$$S = \frac{x+v}{\omega'}$$

$$e^{iv}\hat{\sigma}_b(\omega') = \frac{1}{\omega'}\int_{-\infty}^{\infty}\sigma_b\left(\frac{x+v}{\omega'}\right)e^{-ix}dx$$

Notice that

$$\begin{aligned}\hat{\sigma}_b(\omega') = a(\omega') + ib(\omega') \Rightarrow e^{iv}\hat{\sigma}_b(\omega') &= e^{iv}(a(\omega') + ib(\omega')) \\ &= (\cos(v) + i\sin(v))(a(\omega') + ib(\omega'))\end{aligned}$$

Hence

$$e^{iv}\hat{\sigma}_b(\omega') = (\cos(v)a(\omega') - \sin(v)b(\omega')) + i(\cos(v)b(\omega') + \sin(v)a(\omega'))$$

And (assuming that $a(\omega') \neq 0$)

$$Re\left(e^{iv}\hat{\sigma}_b(\omega')\right) + \frac{b(\omega')}{a(\omega')} Im\left(e^{iv}\hat{\sigma}_b(\omega')\right) = \cos(v)\left(\frac{a^2(\omega') + b^2(\omega')}{a(\omega')}\right)$$

Then

$$\begin{aligned}\cos(v) = &\left(\frac{a(\omega')}{a^2(\omega') + b^2(\omega')}\right) Re\left(\frac{1}{\omega'}\int_{-\infty}^{\infty} \sigma_b\left(\frac{x+v}{\omega'}\right) e^{-ix} dx\right) \\ &+ \left(\frac{b(\omega')}{a^2(\omega') + b^2(\omega')}\right) Im\left(\frac{1}{\omega'}\int_{-\infty}^{\infty} \sigma_b\left(\frac{x+v}{\omega'}\right) e^{-ix} dx\right)\end{aligned}$$

$$Re\left(\frac{1}{\omega'}\int_{-\infty}^{\infty}\sigma_b\left(\frac{x+\upsilon}{\omega'}\right)e^{-ix}dx\right)=\Delta\sum_{k=1}^{N}\sigma_b\left(\frac{x_k+\upsilon}{\omega'}\right)\cos(x_k)+\frac{\varepsilon}{2},\ x_k\in\mathbb{R}$$

$$Im\left(\frac{1}{\omega'}\int_{-\infty}^{\infty}\sigma_b\left(\frac{x+\upsilon}{\omega'}\right)e^{-ix}dx\right)=\Delta\sum_{p=1}^{N}\sigma_b\left(\frac{x_p+\upsilon}{\omega'}\right)\sin(x_p)+\frac{\varepsilon}{2},\ x_p\in\mathbb{R}$$

Hence

$$\cos(\upsilon)=\sum_{j=1}^{2N}c_j\sigma_b(\omega_j\upsilon+b_j)+\varepsilon=\sum_{j=1}^{2N}c_jf(\omega_j\upsilon+b_j)-\sum_{j=1}^{2N}c_jf(\omega_j\upsilon+b_j+b)+\varepsilon$$

and $\bar{f}(\upsilon)$ can be approximated by sums

$$\sum_{i=1}^{M}c_jf(\omega_j\upsilon+b_j)$$

Multi-dimensional case.

Follows from Weierstrass theorem by induction. Two-dimensional approximation involves

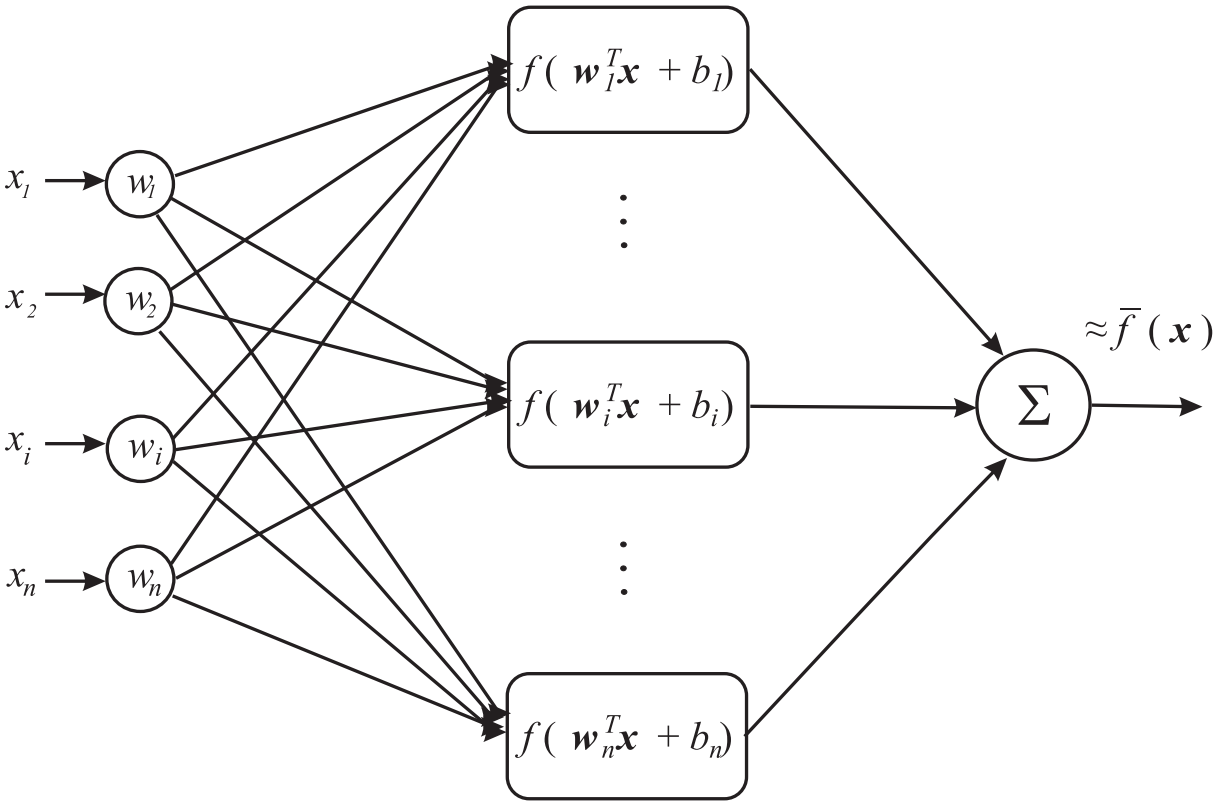
$$\sin(n\upsilon)\sin(m\nu) = \frac{1}{2}(\cos(n\upsilon - m\nu) - \cos(n\upsilon + m\nu))$$

$$\sin(n\upsilon)\cos(m\nu) = \frac{1}{2}(\sin(n\upsilon + m\nu) + \sin(n\upsilon - m\nu))$$

These can be expressed as sums of $\sin(w_1\upsilon + w_2\nu)$, $\cos(w_1\upsilon + w_2\nu)$

$$\Rightarrow \bar{f}(\upsilon, \nu) = \sum_{i=1}^{N} f(w_{1,i}\upsilon + w_{2,i}\nu + b_i) + \varepsilon$$

Finally we have a network !



And the network can fit the data

What about the number of elements in the sums?

$$\bar{f}(\mathbf{x}) = \int_{\mathbb{R}^d} e^{i\omega^T \mathbf{x}}\ \hat{f}(\omega)d\omega,\ C_f = \int_{\mathbb{R}^d} \|\omega\|\|\hat{f}(\omega)\|d\omega,\ B_r = \{\mathbf{x} :\ |\ \|\mathbf{x}\| \leq r\}$$

Theorem (A. Barron, 1993)

For every $\bar{f}$ with C_f finite, and every $n \geq 1$ there exists a linear combination of sigmoidal functions:

$$f_n(\mathbf{x}) = \sum_{i=1}^{n} c_i f(\mathbf{x}^T \mathbf{w}_i + b_i) + c_0, \quad c_i, b_i \in \mathbb{R},\ \mathbf{w}_i \in \mathbb{R}^d$$

such that

$$\int_{B_r} (\bar{f}(\mathbf{x}) - f_n(\mathbf{x}))^2 \leq \frac{(2rC_f)^2}{n}$$

Proof is based on L.K. Jones lemma for greedy approximation (1992)

$$g(\mathbf{x}) \in \{c\ f(\mathbf{w}^T\mathbf{x})\},\ \mathbf{w} \in \mathbb{R}^d,\ c \in \mathbb{R}$$

$$\varphi_{n+1}(\mathbf{x}) = \varphi_n(\mathbf{x})(1-\alpha_n) + \alpha_n g_n(\mathbf{x})$$

If $\bar{f}(\mathbf{x})$ belongs to the closure of convex hull of $\{c\ f(\mathbf{w}^T\mathbf{x})\}$ then the procedure converges as

$$\|\varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x})\| \leq O\left(\frac{1}{\sqrt{n}}\right)$$

(Please find and read L.K. Jones 1993 work)

Main idea

Let $\bar{f}(\mathbf{x})$ belong to a closure of the convex hull of

$$\{f(\mathbf{w}^T\mathbf{x})\beta\},\ \beta \in \mathbb{R},\ |\beta| < D$$

That is:

$$\forall\ \delta > 0\ \exists\ \{\alpha_j, \beta_j, \mathbf{w}_j, b_j\} :\ \|\bar{f}(\mathbf{x}) - \sum_{j=1}^{s} \alpha_j \underbrace{\beta_j f(\mathbf{w}_j^T\mathbf{x} + b_j)}\| \leq \delta$$

$$\sum_{j=1}^{s} \alpha_j = 1,\ 0 \leq \alpha_j \leq 1$$

Claim 1

There exists $\beta_k f(\mathbf{w}_k^T + b_k)$, $|\beta_k| < D$ such that

$$\langle \varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x}), \beta_k f(\mathbf{w}_k^T + b_k) - \bar{f}(\mathbf{x}) \rangle \leq 0$$

Consider

$$\bar{f}(\mathbf{x}) = \sum_{s=1}^{m} \alpha_s \underbrace{\beta_s f(\mathbf{w}_s^T \mathbf{x} + b_s)}, \ \sum_{s=1}^{m} \alpha_s = 1$$

This is always possible because $\bar{f}$ is from the closure of $\{f(\mathbf{w}^T \mathbf{x})\beta\}$. Let

$$\langle \varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x}), \beta_s f(\mathbf{w}_s^T + b_s) - \bar{f}(\mathbf{x}) \rangle > \delta \ \forall \ s \in \{1, 2, \ldots, m\}$$

Then

$$\sum_{s=1}^{m} \alpha_s \langle \varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x}), \beta_s f(\mathbf{w}_s^T + b_s) - \bar{f}(\mathbf{x}) \rangle$$

$$= \langle \varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x}), \sum_{s=1}^{m} \alpha_s \beta_s f(\mathbf{w}_s^T + b_s) - \left(\sum_{s=1}^{m} \alpha_s \right) \bar{f}(\mathbf{x}) \rangle$$

$$= \langle \varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x}), \sum_{s=1}^{m} \alpha_s \beta_s f(\mathbf{w}_s^T + b_s) - \bar{f}(\mathbf{x}) \rangle$$

$$\geq \sum_{s=1}^{m} \alpha_s \delta = \delta$$

On the other hand

$$\sum_{s=1}^{m} \langle \varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x}), \alpha_s \beta_s f(\mathbf{w}_s^T + b_s) - \bar{f}(\mathbf{x}) \rangle = 0$$

Hence we have

$$0 < \delta \leq \sum_{s=1}^{m} \langle \varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x}), \alpha_s \beta_s f(\mathbf{w}_s^T + b_s) - \bar{f}(\mathbf{x}) \rangle \leq \| \varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x}) \| = 0$$

which is a contradiction.

Claim 2

$$\|\varphi_{n+1} - \bar{f}\|^2 \leq M^2 \frac{\|\varphi_n - \bar{f}\|^2}{\|\varphi_n - \bar{f}\|^2 + M^2},$$

$$\|g(\mathbf{x})\| + \|\bar{f}(\mathbf{x})\| \leq M, \quad M > 0$$

Consider

$$\|\varphi_{n+1}(\mathbf{x}) - \bar{f}(\mathbf{x})\|^2 = \|(1-\alpha_n)(\varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x})) + \alpha(g(\mathbf{x}) - \bar{f}(\mathbf{x}))\|^2$$
$$= (1-\alpha_n)^2\|(\varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x})\|^2 + (1-\alpha)\alpha 2\langle\varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x}), g(\mathbf{x}) - \bar{f}(\mathbf{x})\rangle$$
$$+ \alpha^2\|g(\mathbf{x}) - \bar{f}(\mathbf{x})\|^2$$

Hence (see Claim 1)

$$\|\varphi_{n+1}(\mathbf{x}) - \bar{f}(\mathbf{x})\|^2 \leq (1-\alpha_n)^2\|\varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x})\|^2 + \alpha_n^2(M)^2$$

Choosing

$$\alpha_n = \frac{\|\varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x})\|^2}{M^2 + \|\varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x})\|^2},$$

we obtain

$$\|\varphi_{n+1}(\mathbf{x}) - \bar{f}(\mathbf{x})\|^2 \leq \frac{M^4\|\varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x})\|^2}{(M^2 + \|\varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x})\|^2)^2} + \frac{(M)^2\|\varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x})\|^4}{(M^2 + \|\varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x})\|^2)^2}$$

Then

$$\|\varphi_{n+1}(\mathbf{x}) - \bar{f}(\mathbf{x})\|^2 \leq M^2 \left(\frac{\|\varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x})\|^2}{M^2 + \|\varphi_n(\mathbf{x}) - \bar{f}(\mathbf{x})\|^2} \right)$$

Claim 3

$$\left\|\varphi_{n+1}(\mathbf{x}) - \bar{f}(\mathbf{x})\right\|^2 = O\left(\frac{1}{n}\right)$$

Denote $\left\|\varphi_{n+1}(\mathbf{x}) - \bar{f}(\mathbf{x})\right\|^2 = e_{n+1}^2$ and consider

$$\frac{1}{e_{n+1}^2} \geq \frac{1}{M^2}\left(1 + \frac{M^2}{e_n^2}\right) = \frac{1}{M^2} + \frac{1}{e_n^2}$$

Repeating until $n = 0$ we get

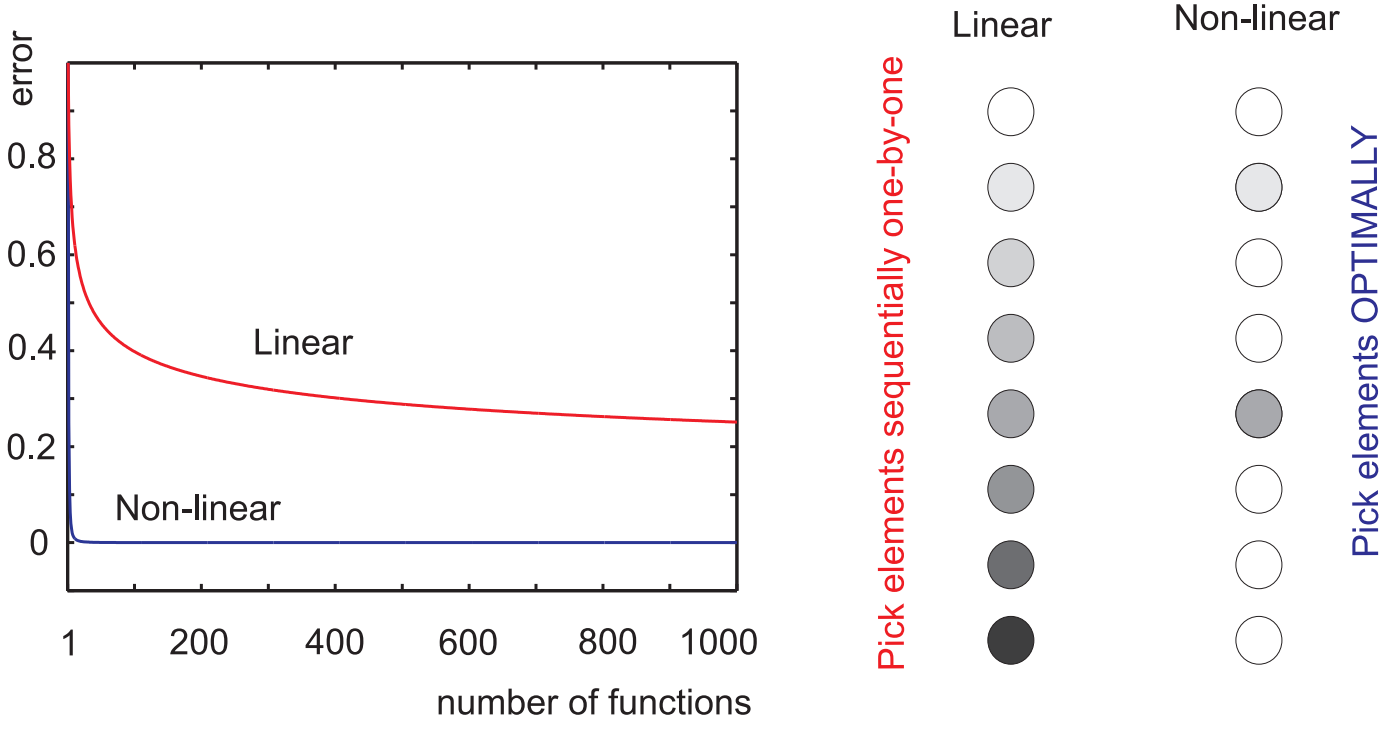
$$\frac{1}{e_{n+1}^2} \geq \frac{1}{M^2}\left(1 + \frac{M^2}{e_n^2}\right) = \frac{1}{M^2} + \frac{1}{e_n^2} \geq \frac{1}{M^2} + \frac{1}{M^2} + \frac{1}{e_{n-1}^2} + \cdots$$
$$+ \cdots \frac{1}{e_0^2} = \frac{n+1}{M^2} + \frac{1}{e_0^2}$$

Therefore

$$e_{n+1}^2 \leq \frac{M^2 e_0^2}{(n+1)e_0^2 + M^2} = O\left(\frac{1}{n+1}\right)$$

This completes the proof of Jones lemma

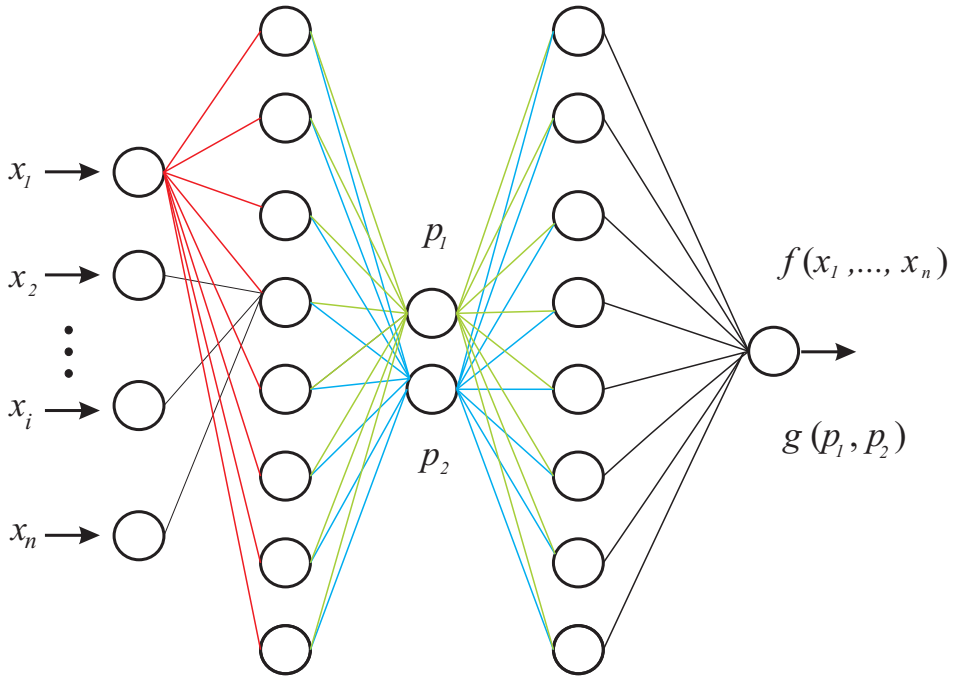
Non-adaptive approximation vs Adaptive, nonlinear choice of functions

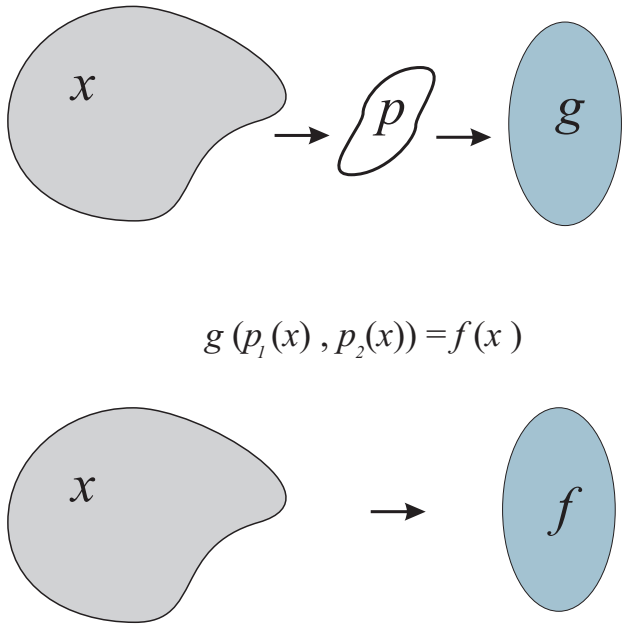


Optimal Convergence rates might be computationally expensive $\Rightarrow$

Tradeoff: Time $_{\text{processing data}}$ vs Time $_{\text{learn optimal representation}}$

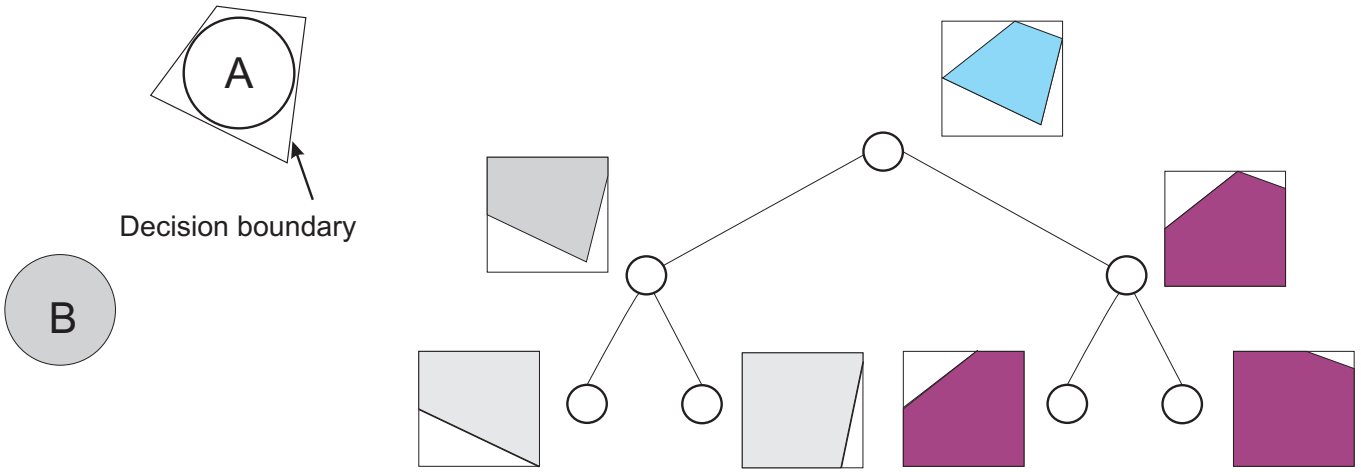
When do we need multi-layer networks





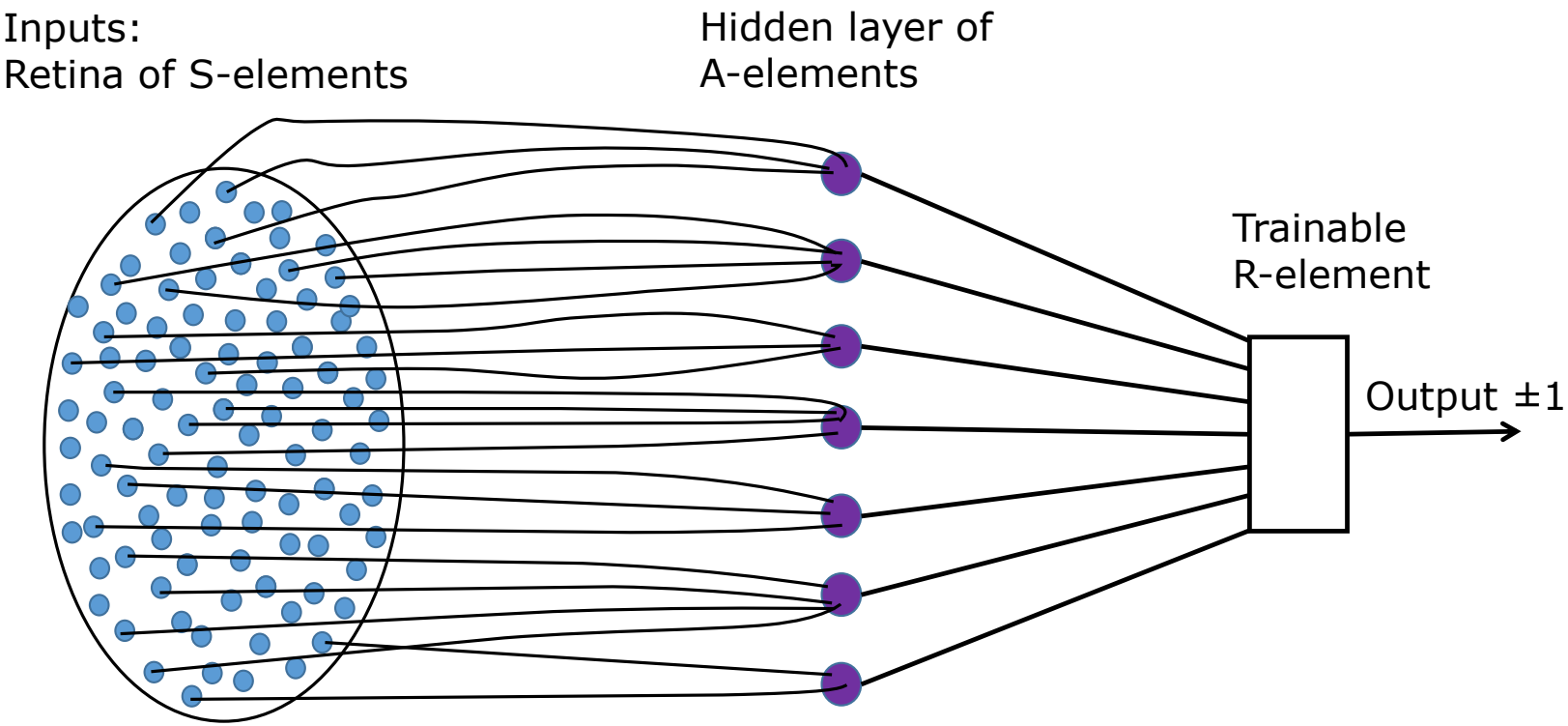
Nonlinear data reduction:

$$f(x_1, \ldots, x_n) = f(p_1(x_1, \ldots, x_n), p_2(x_1, \ldots, x_n))$$



- finding iterative roots;
- complex classification tasks (XOR-type problems in logic) and decision-making.
- and many more

Recall that first Rosenblatt Perceptrons were already deep nets!



For each binary image x we can create an A-element A_x that produces output 1 for this image and 0 for all other.

Indeed, let the input retina have n elements and $x = (x_1, \ldots, x_n)$ be a binary vector ($x_i = 0$ or 1) with k non-zero elements. The corresponding A-element A_x has input synapses with weights $w_i = 1/k$ if $x_i = 1$ and $w_i = -1/k$ if $x_i = 0$. For an arbitrary binary image y,

$$\sum w_i y_i \leq 1$$

and this sum is equal to 1 if and only if $y = x$. The threshold for the A_x output can be selected as $1 - \frac{1}{2k}$. Thus,

$$OutA_x(y) = \begin{cases} 0, & if \ \sum w_i y_i < 1 - \frac{1}{2k}; \\ 1, & if \ \sum w_i y_i \geq 1 - \frac{1}{2k}. \end{cases} \tag{3}$$

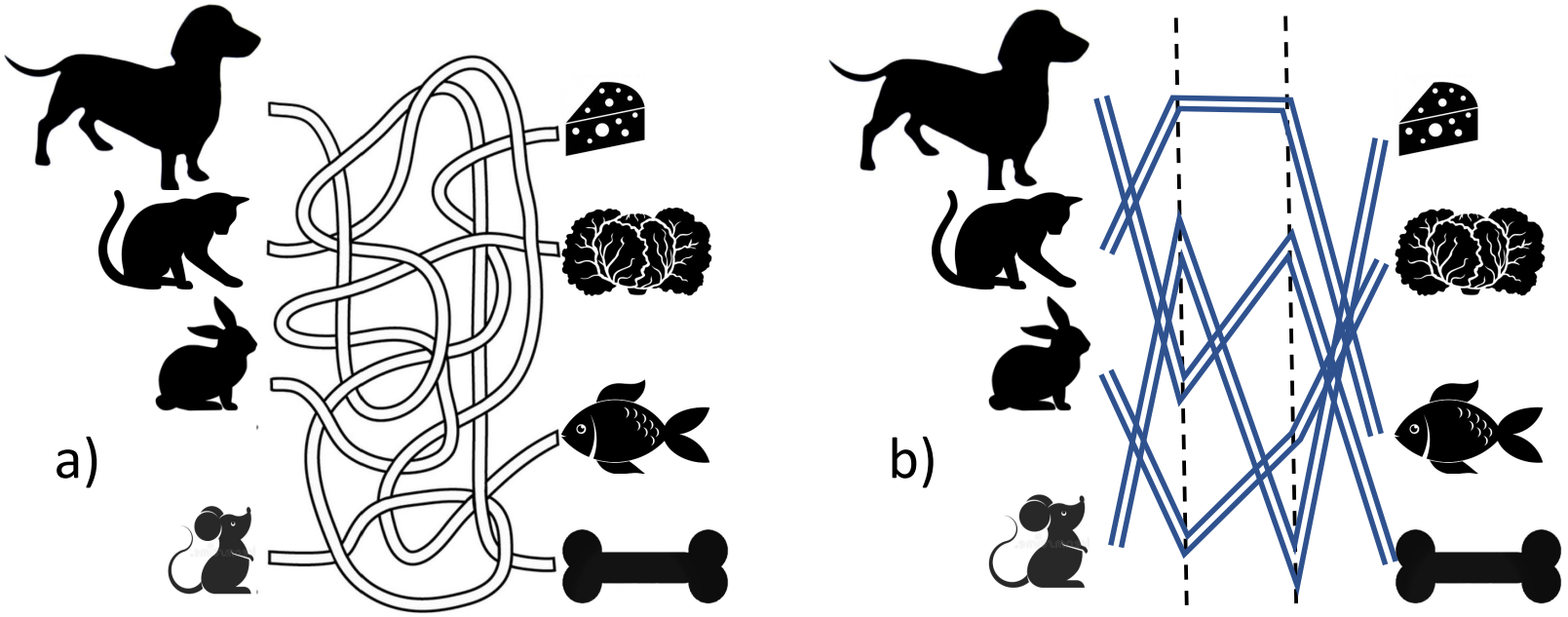
The set of neurons A_x created for all binary vectors x transforms binary images into the vertexes of the standard simplex in $\mathbb{R}^{2^n}$ with coordinates $OutA_x(y)$ (3). Any two non-intersecting subsets of the standard simplex can be separated by a hyperplane. Therefore, there exists an R-element that separates them. According to the perceptron convergence theorem, this R-element can be found by the perceptron learning algorithm.

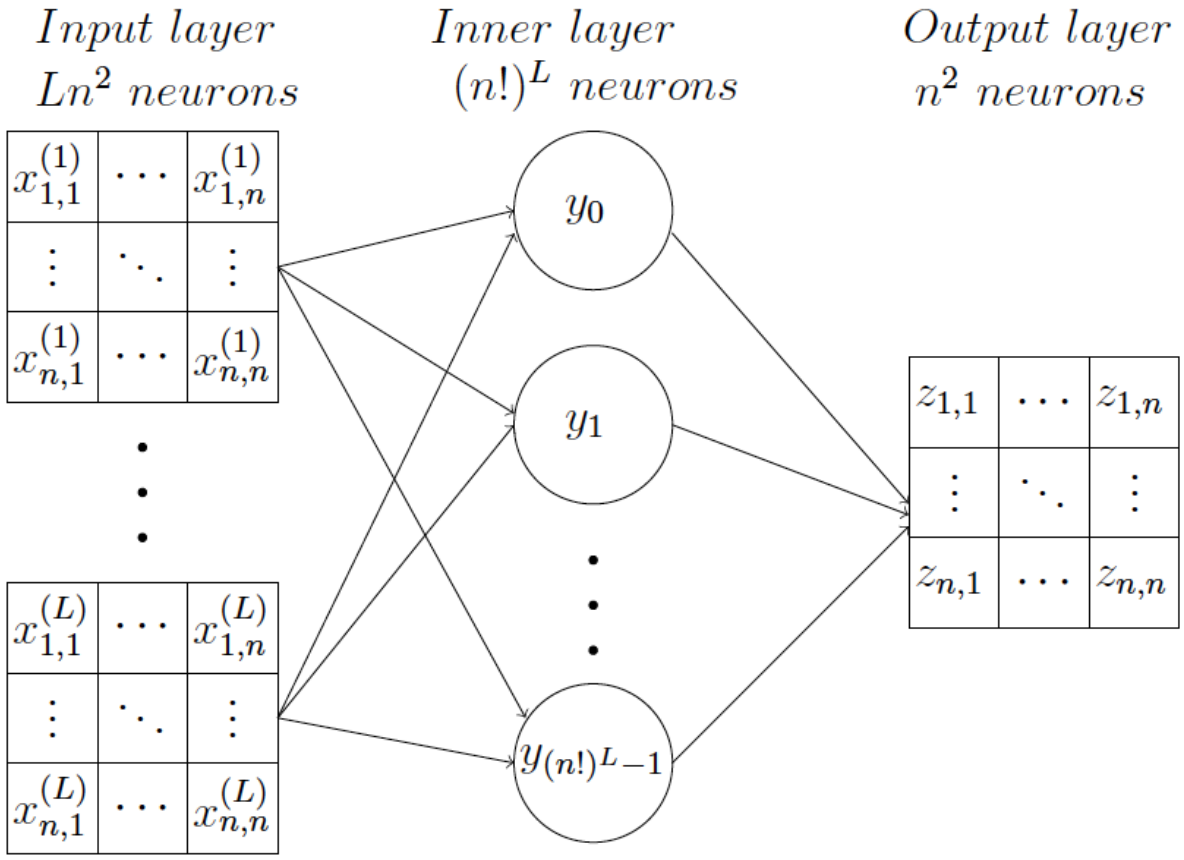
Theorem

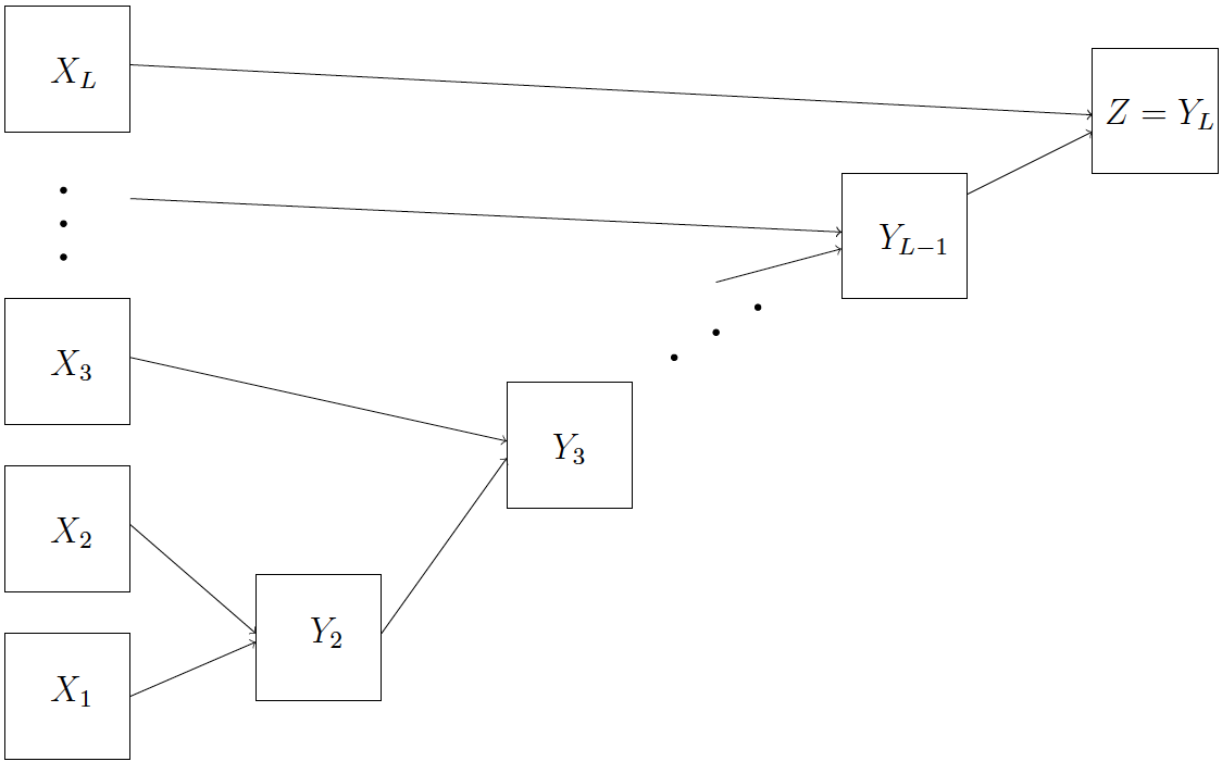
Elementary perceptron can separate any two non-intersecting sets of binary images.

Problems: One needs exponentially many elements $\sim 2^n$ to solve this task.

Another problem:




Input layer
Ln^2 neurons
Inner layer
$(n!)^L$ neurons
Output layer
n^2 neurons
$x_{1,1}^{(1)}$ $\cdots$ $x_{1,n}^{(1)}$
$x_{n,1}^{(1)}$ $\cdots$ $x_{n,n}^{(1)}$
$x_{1,1}^{(L)}$ $\cdots$ $x_{1,n}^{(L)}$
$x_{n,1}^{(L)}$ $\cdots$ $x_{n,n}^{(L)}$
y_0
y_1
$y_{(n!)^L-1}$
$z_{1,1}$ $\cdots$ $z_{1,n}$
$z_{n,1}$ $\cdots$ $z_{n,n}$


X_L
$Z = Y_L$
Y_{L-1}
X_3
Y_3
X_2
Y_2
X_1

(see: https://arxiv.org/pdf/2208.13778 for details)

Theorem

The constructed shallow neural network has a depth of 3,

$$(L+1)n^2 + (n!)^L$$

neurons, and

$$(L+1)n^2(n!)^L$$

connections between neurons.

Theorem

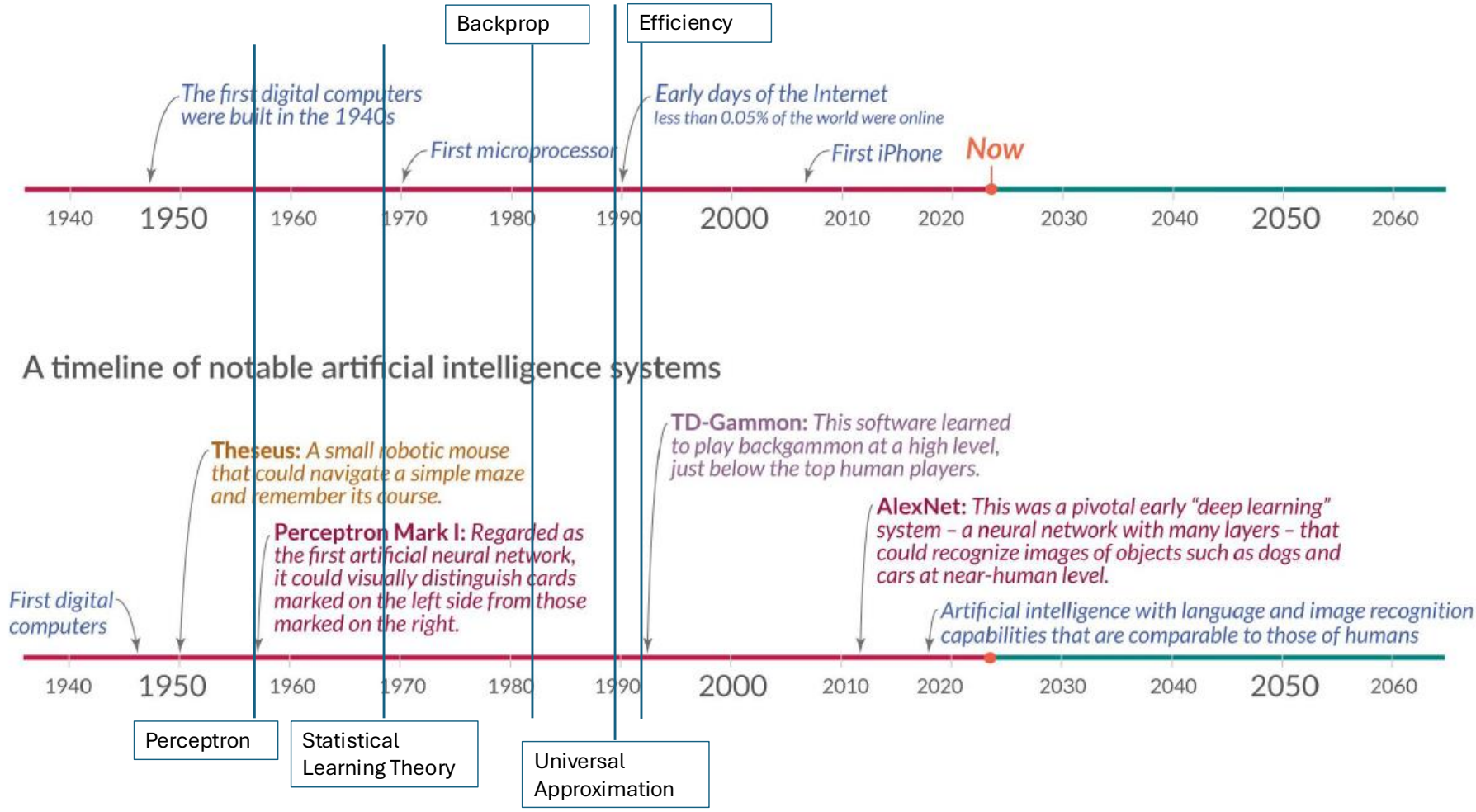
The constructed deep neural network has a depth of L,

$$(2L-1)n^2$$

neurons, and

$$2(L-1)n^3$$

connections between neurons.


Backprop
Efficiency
The first digital computers were built in the 1940s
First microprocessor
Early days of the Internet
less than 0.05% of the world were online
First iPhone
Now
1940
1950
1960
1970
1980
1990
2000
2010
2020
2030
2040
2050
2060
A timeline of notable artificial intelligence systems
Theseus: A small robotic mouse that could navigate a simple maze and remember its course.
Perceptron Mark I: Regarded as the first artificial neural network, it could visually distinguish cards marked on the left side from those marked on the right.
TD-Gammon: This software learned to play backgammon at a high level, just below the top human players.
AlexNet: This was a pivotal early "deep learning" system – a neural network with many layers – that could recognize images of objects such as dogs and cars at near-human level.
First digital computers
Artificial intelligence with language and image recognition capabilities that are comparable to those of humans
Perceptron
Statistical Learning Theory
Universal Approximation

- Solving ODEs: $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta f(\mathbf{x}_k)$
- Various multi-stage processes

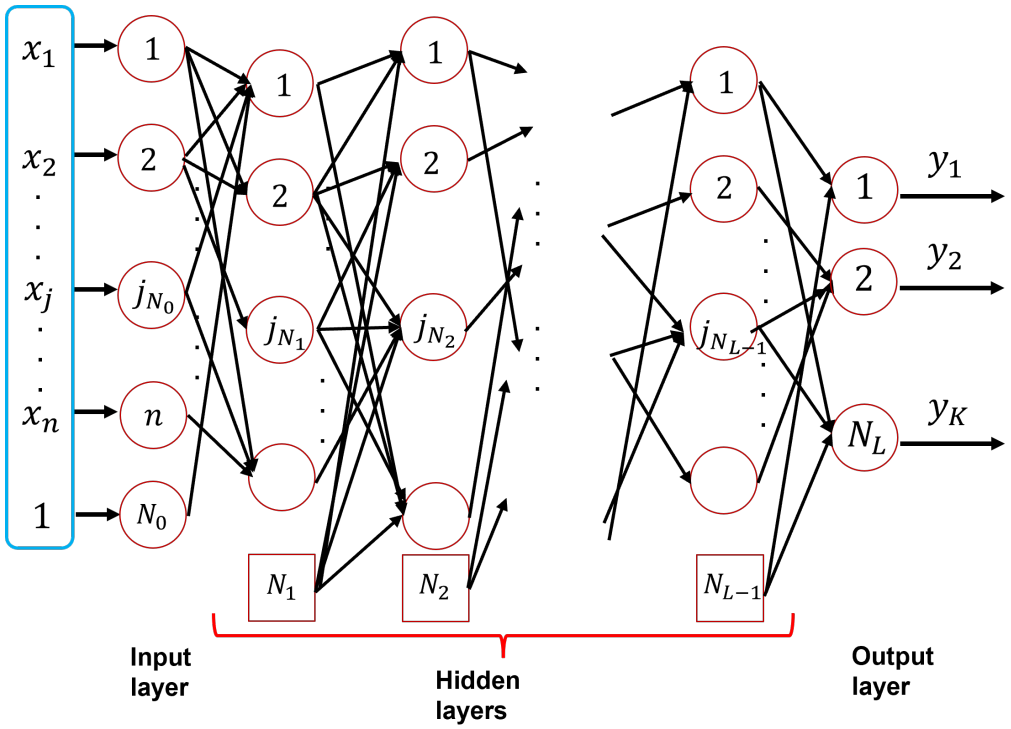
What is then a deep learning?

Deep Learning is a branch of Machine Learning focussing, developing, and implementing end-to-end solutions of various machine learning tasks using Multilayer Neural Networks.

In contrast with classical machine learning, deep learning does not require hand-crafted and meticulously chosen features. Instead it determines relevant features automatically directly from data.

Multilayer Neural Networks come in different shapes and names:

- MLP - multilayer perceptron
- DNN - Deep neural networks (with a mixture of activations and layers)
- RNN - Recurrent neural networks
- CNN - Convolutional neural networks
- LSTM - Long short term memory
- Autoencoders etc



$$q_j^m(q_1^{m-1},\ldots,q_{N_{m-1}}^{m-1}) = f\left(\sum_{i=1}^{N_{m-1}} w_{j,i}^m q_i^{m-1} + b_j^m\right)$$

where N_{m-1} - number of units in layer $m-1$, and q_j^m is the "output" of the j-th unit in layer m, $m = 1, 2, ..., L$.

- Let $\mathbf{x} = \mathrm{col}(x_1, \ldots, x_n)$ be an input vector from $\mathbf{X} \subset \mathbf{R}^n$
- Let $\mathbf{y}(\mathbf{x}) = \mathrm{col}(y_1(\mathbf{x}), \ldots, y_K(\mathbf{x}))$ be a mapping from $\mathbf{X}$ to $\mathbf{Y} \subset \mathbf{R}^K$
- Let $\mathbf{q}^L(\mathbf{w}, \mathbf{x})$ be the network output ($\dim(\mathbf{q}^L) = K$)
- Let $E(\mathbf{x}, \mathbf{q}^L(\mathbf{w}, \mathbf{x}))$ be a cost function. For example,

$$E(\mathbf{x}, \mathbf{q}^L(\mathbf{w}, \mathbf{x})) = \frac{1}{2} \sum_{i=1}^{N_L} (\mathbf{y}(\mathbf{x}_i) - \mathbf{q}^L(\mathbf{x}_i, \mathbf{w}))^2$$

We wish to find $\mathbf{w}^*$ (with b_j-s included into $\mathbf{w}$ as components):

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} E(\mathbf{x}, \mathbf{q}^L(\mathbf{w}, \mathbf{x}))$$

This is a hard problem!

One way to approach the problem is to use the gradient descent algorithm:

$$\Delta \mathbf{w} = -\gamma \frac{\partial E(\mathbf{x}, \mathbf{q}^K(\mathbf{w}, \mathbf{x}))}{\partial \mathbf{w}}$$

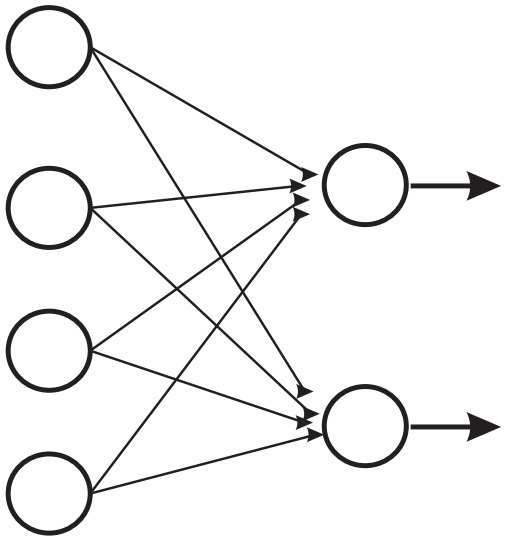
Naive estimate of computational costs

- Let N be the dimension of $\mathbf{w}$
- Naive (direct) calculation of

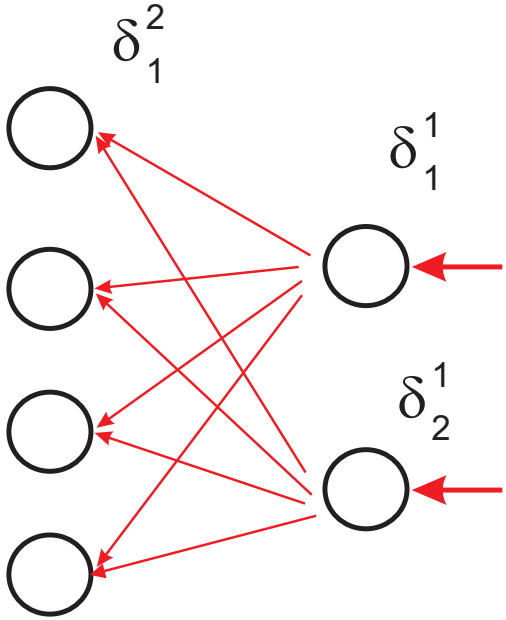
$$\frac{\partial E(\mathbf{x}, \mathbf{q}^L(\mathbf{w}, \mathbf{x}))}{\partial \mathbf{w}_j}, \ j \in \{1, \ldots, N\}$$

  is of the order of N for each individual component of $\mathbf{w}$ (since in the worst case we would have to evaluate a function parameterized by at least N parameters)
- We have N components, hence it looks like the costs are $O(N^2)$

But we can do better, thanks to the chain rule.



Forward



Backward

Just two passes through the network and its "dual", hence complexity of computing all gradients is $O(N)$

Consider

$$\frac{\partial E(\mathbf{x}, \mathbf{q}^L(\mathbf{w}, \mathbf{x}))}{\partial \mathbf{w}_j^L} = \frac{\partial E(\mathbf{x}, \mathbf{q}^L(\mathbf{w}, \mathbf{x}))}{\partial q_j^L} \frac{\partial q_j^L}{\partial \mathbf{w}_j^L}, \ j \in \{1, \ldots, N_L\}$$

$$\frac{\partial q_j^L}{\partial \mathbf{w}_j^L} = \frac{\partial f(S_j^L)}{\partial S_j^L} \frac{\partial S_j^L}{\partial \mathbf{w}_j^L} = \frac{\partial f(S_j^L)}{\partial S_j^L} \mathbf{q}^{L-1}$$

Let

$$\delta_j^1 = \frac{\partial E(\mathbf{x}, \mathbf{q}^L(\mathbf{w}, \mathbf{x}))}{\partial q_j^L} = \frac{\partial}{\partial q_j^L} \left(\frac{1}{2} \sum_{i=1}^{N_L} (y(\mathbf{x}_i) - q_j^L(\mathbf{x}_i, \mathbf{w}))^2 \right)$$

$$= -\sum_{i=1}^{N_L} (y(\mathbf{x}_i) - q_j^L(\mathbf{x}_i, \mathbf{w}))$$

Next we consider $L-1$ layer:

$$\frac{\partial E(\mathbf{x}, \mathbf{q}^L(\mathbf{w}, \mathbf{x}))}{\partial \mathbf{w}_j^{L-1}} = \frac{\partial E(\mathbf{x}, \mathbf{q}^L(\mathbf{w}, \mathbf{x}))}{\partial q_j^{L-1}} \frac{\partial q_j^{L-1}}{\partial \mathbf{w}_j^{L-1}}, \; j \in \{1, \ldots, N_{L-1}\}$$

$$\frac{\partial q_j^{L-1}}{\partial \mathbf{w}_j^{L-1}} = \frac{\partial f(S_j^{L-1})}{\partial S_j^{L-1}} \frac{\partial S_j^{L-1}}{\partial \mathbf{w}_j^{L-1}} = \frac{\partial f(S_j^{L-1})}{\partial S_j^{L-1}} \mathbf{q}^{L-2},$$

where $S_j^{L-1} = \sum_{i=1}^{N_{L-1}} (w_{i,j}^L q_i^{L-1})$

$$\delta_j^2 = \frac{\partial E(\mathbf{x}, \mathbf{q}^L(\mathbf{w}, \mathbf{x}))}{\partial q_j^{L-1}} = \frac{\partial}{\partial q_j^{L-1}} E(\mathbf{x}, \mathrm{col}(q_1^L(\mathbf{w}, \mathbf{x}), \ldots, q_{N_L}^L(\mathbf{w}, \mathbf{x})))$$

$$= \sum_{i=1}^{N_L} \frac{\partial E}{\partial q_i^L} \frac{\partial q_i^L(\mathbf{w}, \mathbf{x})}{\partial q_j^{L-1}} = \sum_{i=1}^{N_L} \delta_i^1 \frac{\partial q_i^L(\mathbf{w}, \mathbf{x})}{\partial q_j^{L-1}} = \sum_{i=1}^{N_L} \delta_i^1 \left[\frac{\partial f(S_i^1)}{\partial S_i^1} \right] w_{j,i}^L$$

$$\boxed{\delta_j^2 = \sum_{i=1}^{N_L} \delta_i^1 \left[\frac{\partial f(S_j^L)}{\partial S_j^L} \right] w_{j,i}^L}$$

Following the same logic, we have that

$$\delta_j^{L-t+1} = \frac{\partial E(\mathbf{x}, \mathbf{q}^L(\mathbf{w}, \mathbf{x}))}{\partial q_j^t} = \frac{\partial}{\partial q_j^t} E(\mathbf{x}, \text{col}(q_1^L(\mathbf{w}, \mathbf{x}), \ldots, q_{N_L}^L(\mathbf{w}, \mathbf{x})))$$

$$= \sum_{i=1}^{N_{t+1}} \frac{\partial E}{\partial q_i^{t+1}} \frac{\partial q_i^{t+1}(\mathbf{w}, \mathbf{x})}{\partial q_j^t} = \sum_{i=1}^{N_{t+1}} \delta_i^{L-t} \frac{\partial q_i^{t+1}(\mathbf{w}, \mathbf{x})}{\partial q_j^t}$$

$$\boxed{\delta_j^{L-t+1} = \sum_{i=1}^{N_{t+1}} \delta_i^{L-t} \frac{\partial q_i^{t+1}(\mathbf{w}, \mathbf{x})}{\partial q_j^t} = \sum_{i=1}^{N_{t+1}} \delta_i^{L-t} \left[\frac{\partial f(S_i^{t+1})}{\partial S_i^{t+1}} \right] w_{j,i}^{t+1}}$$

For layers $m \geq 1$ with nonlinear activation functions

$$q_j^m(q_1^{m-1}, \ldots, q_{N_{m-1}}^{m-1}) = f\left(\sum_{i=1}^{N_{m-1}} w_{j,i}^m q_i^{m-1} \right)$$

$$\boxed{\delta_j^{K-m+1} = \sum_{i=1}^{N_{m+1}} w_{i,j}^{m+1} \delta_i^{K-m} \left[\left. \frac{\partial f(S_i^{m+1})}{\partial S_i^{m+1}} \right|_{S_i^{m+1} = \sum_{p=1}^{N_m} (w_{i,p}^{m+1} q_p^m)} \right]}$$

Finally, we can write the increment weights for all layers $t \geq 2$:

$$\Delta w_{i,j}^t = -\gamma \frac{\partial E(\mathbf{x}, \mathbf{q}^L(\mathbf{w}, \mathbf{x}))}{\partial q_j^t} \frac{\partial q_j^t}{\partial w_{i,j}^t}$$

$$= -\gamma \delta_p^{L-t+1} \left[\frac{\partial f(S_j^t)}{\partial S_j^t} \right] q_i^{t-1}$$

$$\mathbf{w}_{j,new}^t = \mathbf{w}_j^t + \Delta \mathbf{w}_j^t$$

and for layer 1:

$$\Delta w_{i,j}^1 = -\gamma \frac{\partial E(\mathbf{x}, \mathbf{q}^L(\mathbf{w}, \mathbf{x}))}{\partial q_j^1} \frac{\partial q_j^1}{\partial w_{i,j}^1}$$

$$= -\gamma \delta_i^L \left[\frac{\partial f(S_i^t)}{\partial S_i^t} \right] x_i$$

$$\mathbf{w}_{j,new}^1 = \mathbf{w}_j^1 + \Delta \mathbf{w}_j^1$$

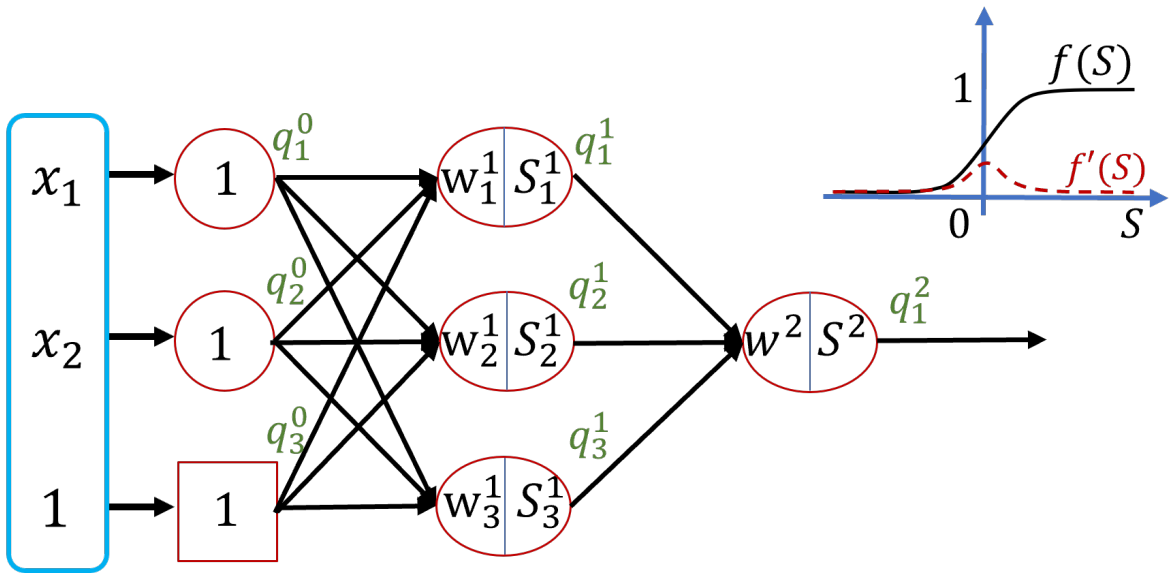
Problem

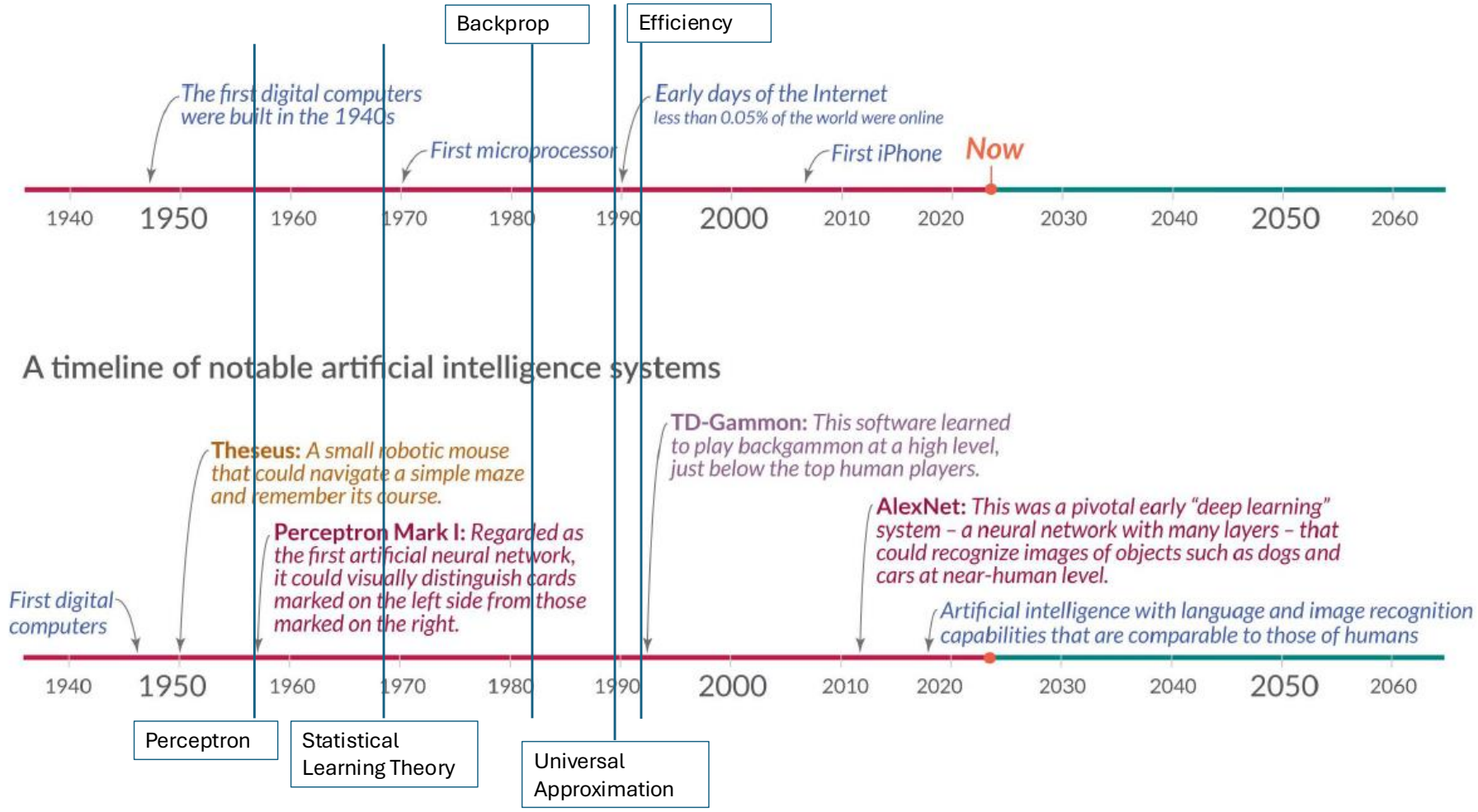
Consider the following data set

x_1	x_2	*Class*	x_1	x_2	*Class*
1	*2*	*1*	*3*	*1*	*0*
3	*3*	*1*	*4*	*2*	*0*

Classify data using the error backpropagation algorithm.

Solution:


x_1
x_2
1
1
1
1
q_1^0
q_2^0
q_3^0
w_1^1 S_1^1
w_2^1 S_2^1
w_3^1 S_3^1
q_1^1
q_2^1
q_3^1
w^2 S^2
q_1^2
$f(S)$
$f'(S)$
1
0
S


Backprop
Efficiency
The first digital computers were built in the 1940s
First microprocessor
Early days of the Internet
less than 0.05% of the world were online
First iPhone
Now
1940
1950
1960
1970
1980
1990
2000
2010
2020
2030
2040
2050
2060
A timeline of notable artificial intelligence systems
Theseus: A small robotic mouse that could navigate a simple maze and remember its course.
Perceptron Mark I: Regarded as the first artificial neural network, it could visually distinguish cards marked on the left side from those marked on the right.
TD-Gammon: This software learned to play backgammon at a high level, just below the top human players.
AlexNet: This was a pivotal early "deep learning" system – a neural network with many layers – that could recognize images of objects such as dogs and cars at near-human level.
First digital computers
Artificial intelligence with language and image recognition capabilities that are comparable to those of humans
Perceptron
Statistical Learning Theory
Universal Approximation