# Kybernetika a umělá inteligence

---

## 4. Shlukovani a neuronové sítě


Gerstnerova laboratoř
katedra kybernetiky
České vysoké učení technické v Praze

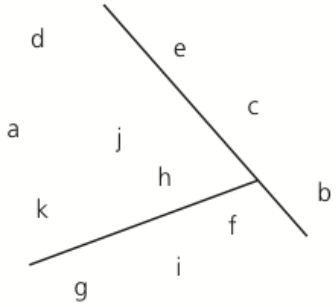
Daniel Novák

## Shrnutí minulé přednášky

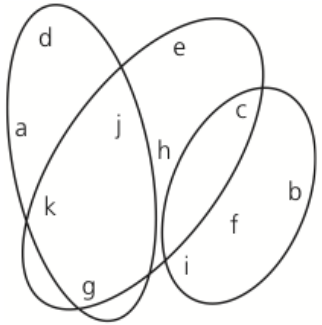
- Známe pravděpodobností rozložení a jeho parametry. Dosadím do Bayesova vzorce a maximalizuji aposteriorní pravděpodobnost, např. $\arg\max_s p(s|x, \mu_s, \sigma_s)$
- V případě normálního případu dostávám kvadratickou diskriminační funkci, v prípadě rovnosti kovariancí lineární diskriminační funkci
- Odhad parametrů pomocí metody maximální věrohodnosti
- Zavedení lineární diskriminační funkce (NEZNÁM PRAVDĚPODOBNOST), odhad parametrů vede na perceptonový algoritmus
- Nulová chyba klasifikace pouze pro lineárně separabilní problém
- Nelineární separabilní problém
  - Transforamce dat do prostoru vyšší dimenze, např. přidáním kvadratických členů
  - Použití sofistikovanějších klasifikátorů, např.neuronové sítě
  - Rozhododovaci stromy: diskriminacni hranice, konstrukce, mira entropie

# Shlukovaní

- Nemam k dispozici trenovaci mnozinu
- Hledam v datech prirozene shluky
- (a) k-means, (b) fuzzy shlukovani, (c) pravdebodobnosti, napr. Gaussovska smes (EM algortimus), (d) hierarchicke shlukovani (dendogram)



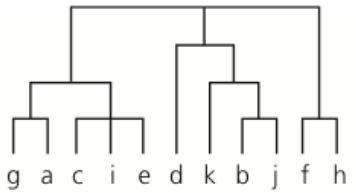
(a)



(b)

| | 1 | 2 | 3 |
|---|---|---|---|
| a | 0.4 | 0.1 | 0.5 |
| b | 0.1 | 0.8 | 0.1 |
| c | 0.3 | 0.3 | 0.4 |
| d | 0.1 | 0.1 | 0.8 |
| e | 0.4 | 0.2 | 0.4 |
| f | 0.1 | 0.4 | 0.5 |
| g | 0.7 | 0.2 | 0.1 |
| h | 0.5 | 0.4 | 0.1 |

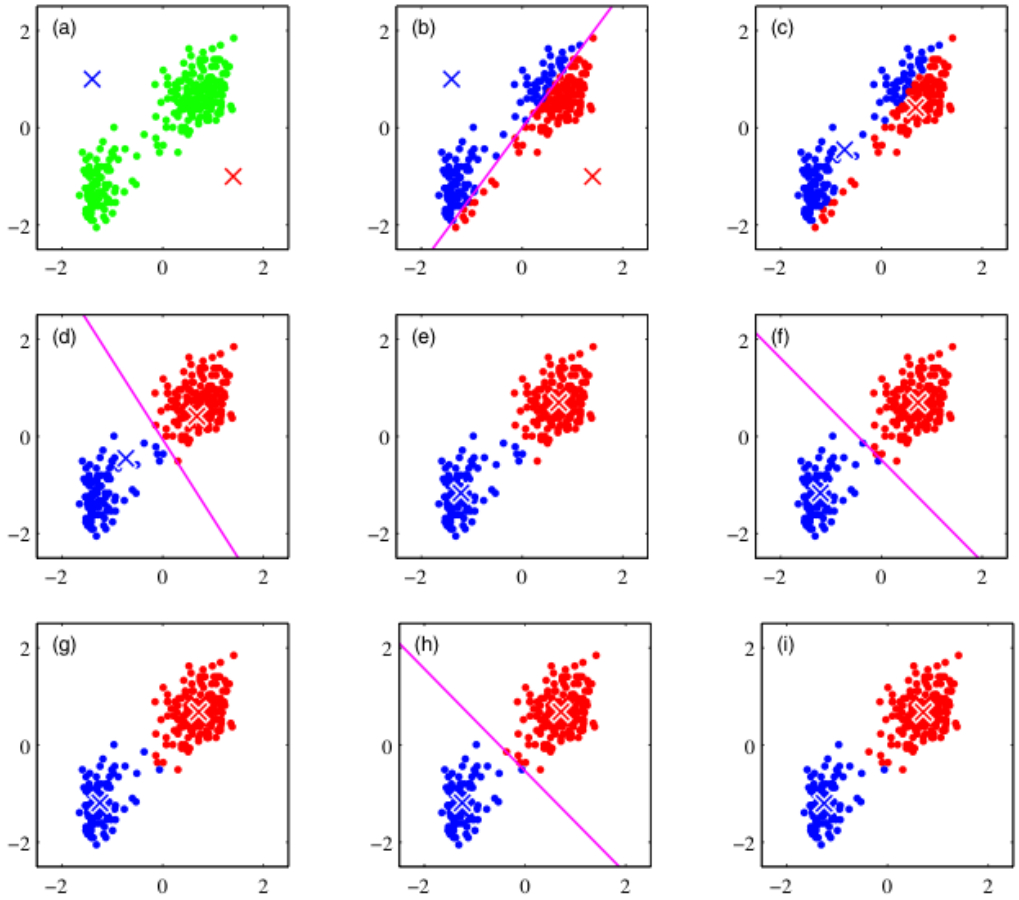
(c)



(d)

# K-means

1. begin inicializuj $k, \mu_1, \mu_2, \ldots, \mu_k$
2. do klasifikuj vzorek podle nejblizsiho $\mu_i$
3. prepocitej $\mu_i$
4. until zadna zmena $\mu_i$
5. return $\mu_1, \mu_2, \ldots, \mu_k$
6. end



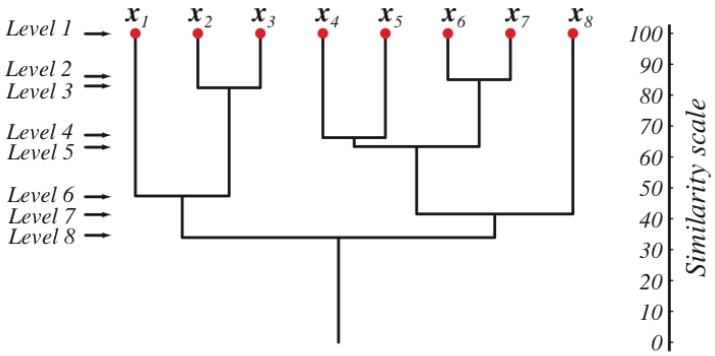
# Hierarchicke shlukovani

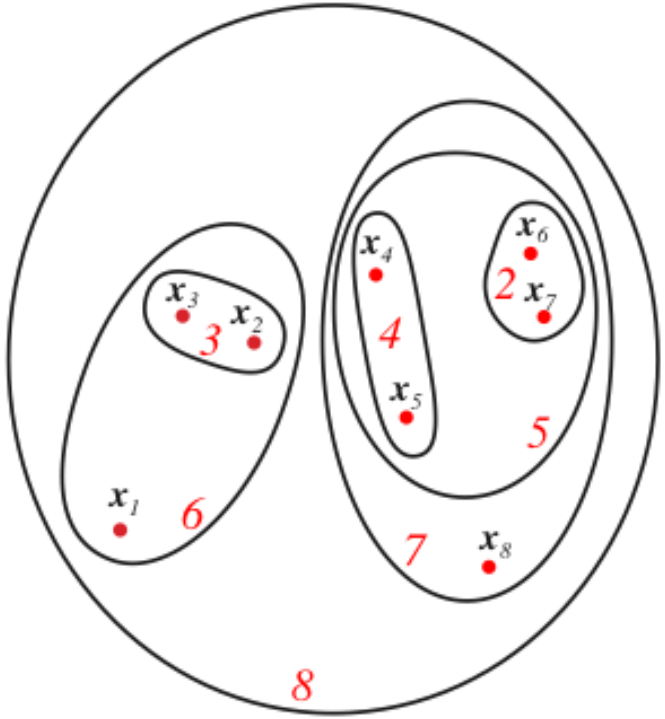
- agglomerative: bottom-up $\rightarrow$ merging
- divisive: top-down $\rightarrow$ splitting

1. begin inicializuj $k, \hat{k} \leftarrow n, \mathcal{D}_i \leftarrow \{X_i\}, i = 1, \ldots, n$
2. do $\hat{k} = \hat{k} - 1$
3. Najdi nejblizsi shluky, napr. $\mathcal{D}_i$ a $\mathcal{D}_j$
4. until $k = \hat{k}$
5. return $k$ shluku
6. end

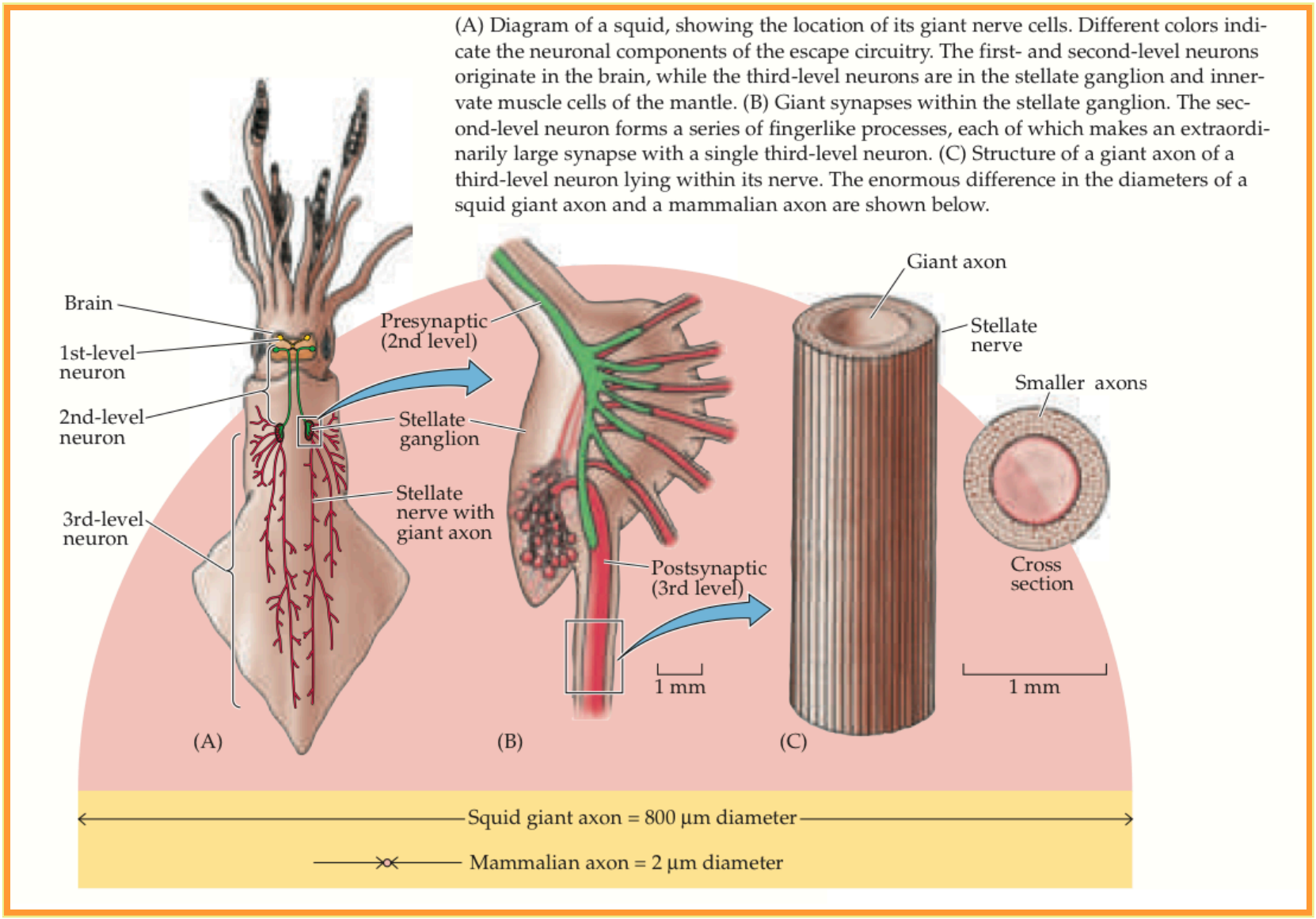
- $d_{min}(x, x^{'}) = \min \|x - x^{'}\|$, $x \in \mathcal{D}_i$, $x^{'} \in \mathcal{D}_i$

# Hierarchicke shlukovani - priklad

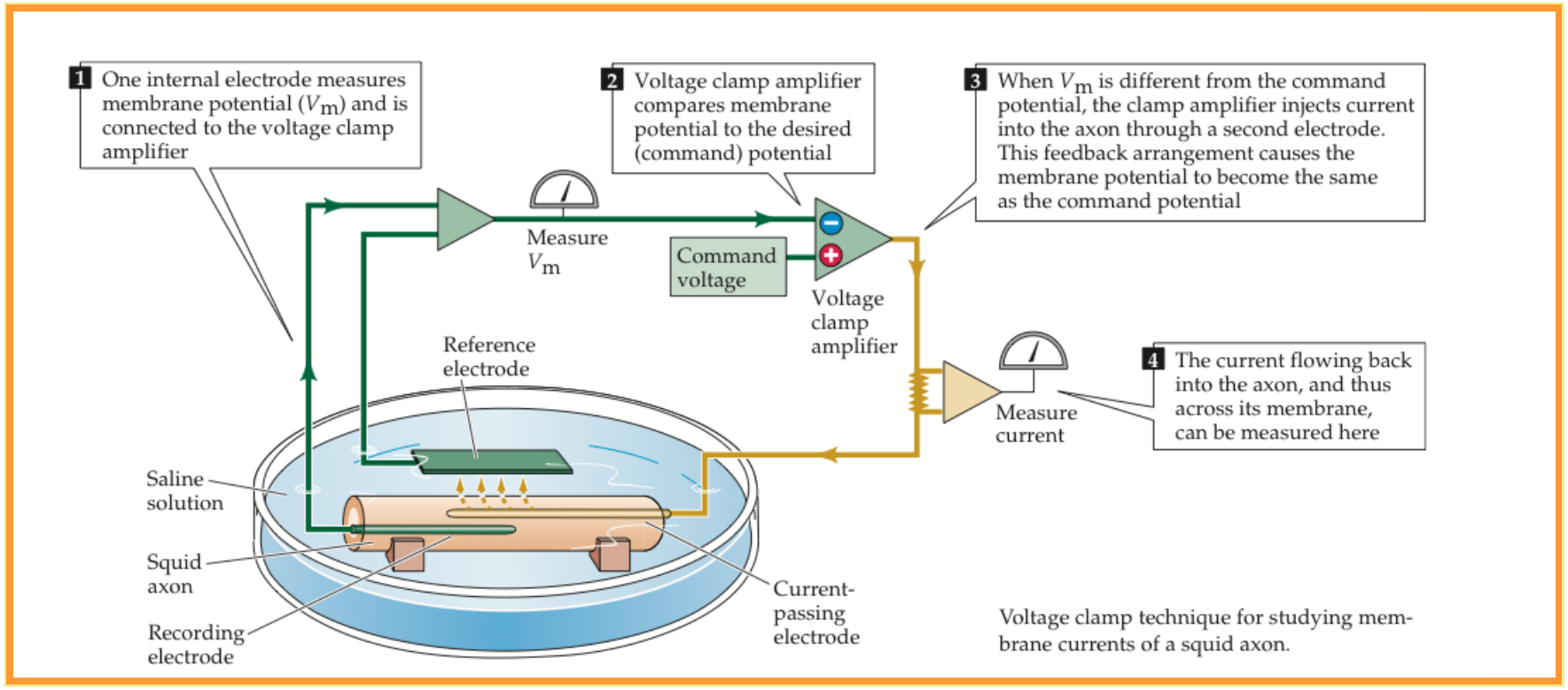


# [Giant Nerve Cells of Squid]

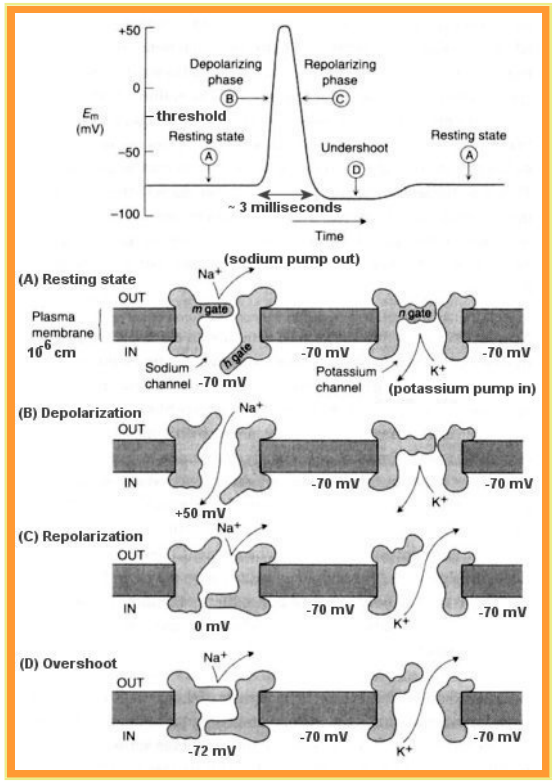


# [Voltage Clamp Method]



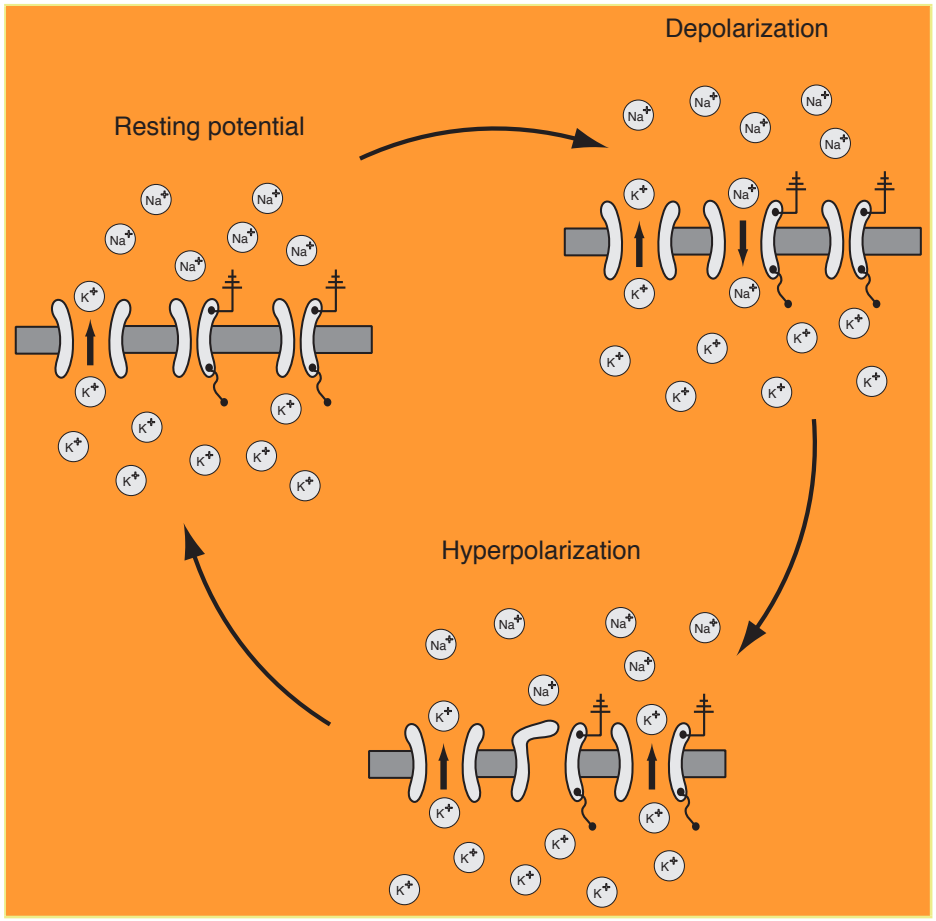

Voltage clamp technique for studying membrane currents of a squid axon.

# [Hodgkin–Huxley model]

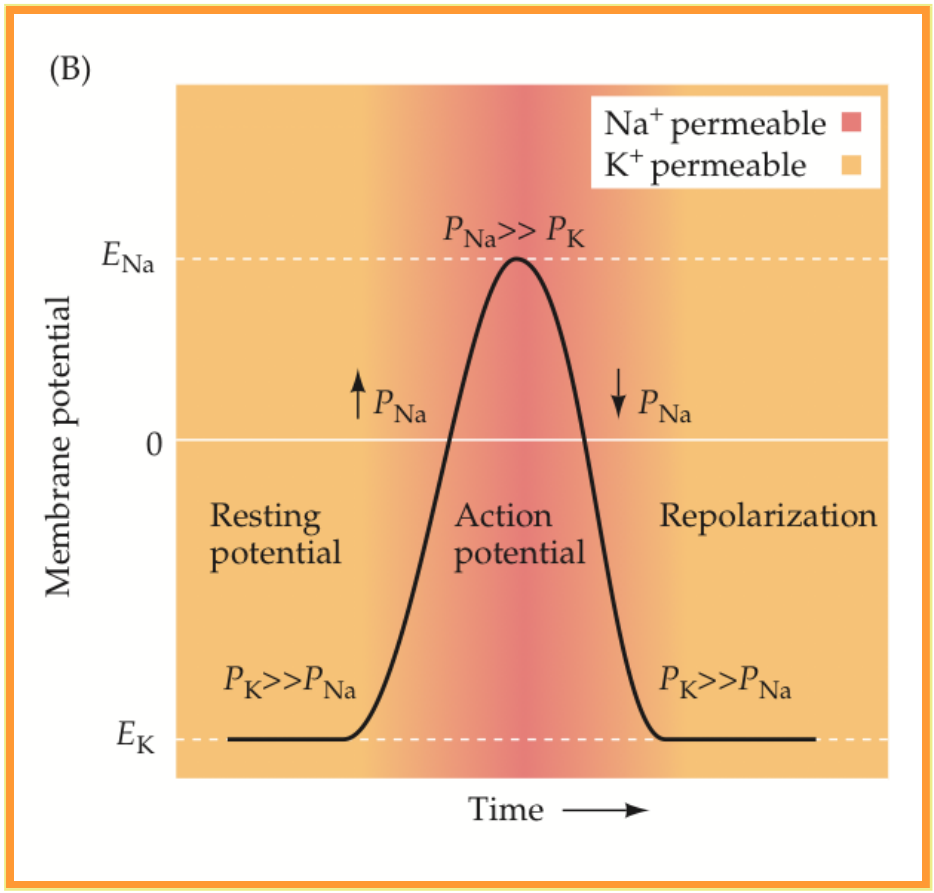



Obrázek 1: Typical form of an action potential; redrawn from an oscilloscope picture from Hodgkin and Huxley (1939).

# [The minimal mechanisms]



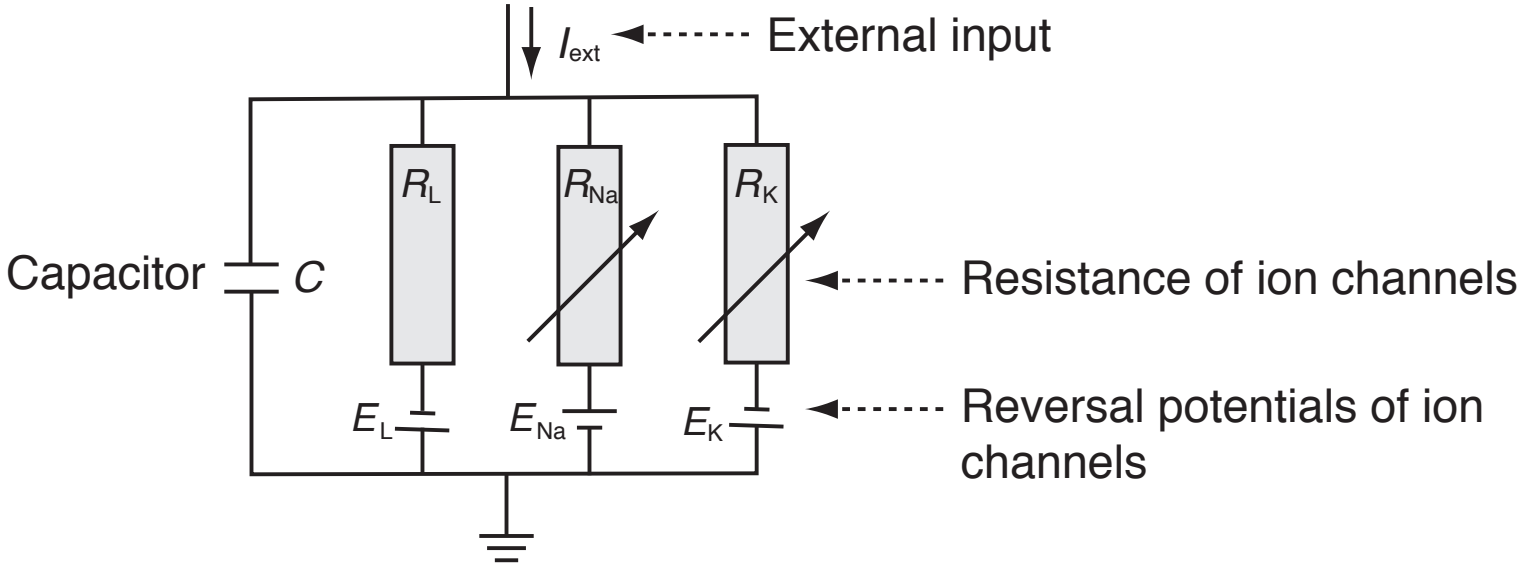
# [Concentration of Na,K]



# [HH stucture]

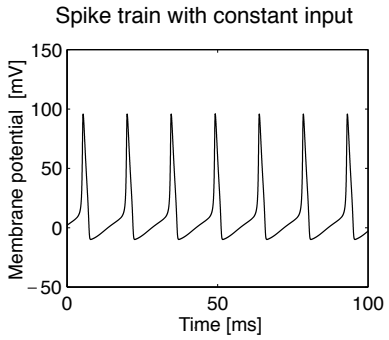
- $I_{ion} = \hat{g_{ion}}(V - E_{ion})$
- voltage and time dependent variables $n(V,t)$,$m(V,t)$,$h(V,t)$

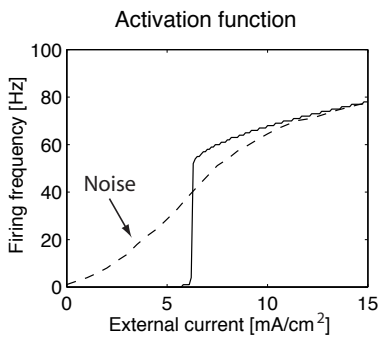
$$\hat{g_K}(V,t) = g_K n^4$$
$$\hat{g_{Na}}(V,t) = g_{Na} m^3 h$$



# [Hodgkin–Huxley equations and simulation]

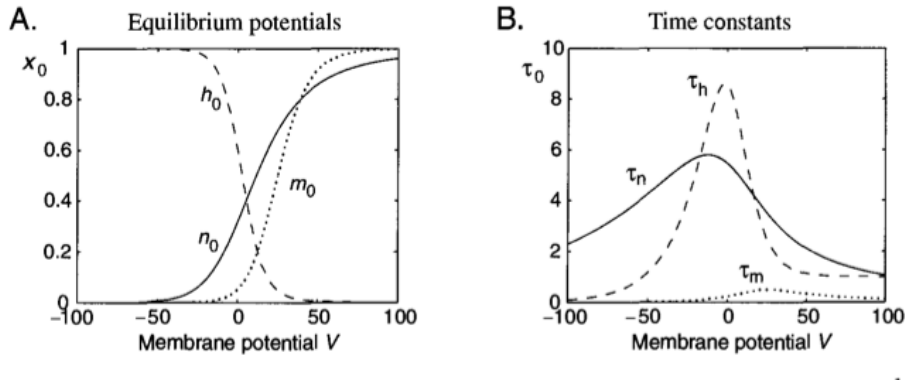
$$
\begin{aligned}
C\frac{\mathrm{d}V}{\mathrm{d}t} &= -g_{\mathrm{K}}n^4(V-E_{\mathrm{K}}) - g_{\mathrm{Na}}m^3h(V-E_{\mathrm{Na}}) - g_{\mathrm{L}}(V-E_{\mathrm{L}}) + I_{ext}(t) \\
\tau_{\mathrm{n}}(V)\frac{\mathrm{d}n}{\mathrm{d}t} &= -[n-n_0(V)] \\
\tau_{\mathrm{m}}(V)\frac{\mathrm{d}m}{\mathrm{d}t} &= -[m-m_0(V)] \\
\tau_{\mathrm{h}}(V)\frac{\mathrm{d}h}{\mathrm{d}t} &= -[h-h_0(V)] \\
\frac{dx}{dt} &= -\frac{1}{\tau_x(V)}[x-x_0(V)] \rightarrow x(t+\Delta t) = (1-\frac{\Delta t}{\tau_x})x(t) + \frac{\Delta t}{\tau_x}x_0
\end{aligned}
$$



# [Ion channels resistance]

$$
\begin{aligned}
x(0) &= \frac{\alpha}{\alpha+\beta}, t_x = \alpha\beta, x \in= \{n, m, h\} \\
\alpha_n &= \frac{10-V}{100(e^{\frac{10-V}{10}}-1)}, \beta_n = 0.125e^{-\frac{V}{80}} \\
\alpha_m &= \frac{25-V}{10(e^{\frac{25-V}{10}}-1)}, \beta_m = 4e^{-\frac{V}{18}} \\
\alpha_h &= 0.07e^{\frac{V}{20}}, \beta_h = \frac{1}{e^{\frac{30-V}{10}}+1}
\end{aligned}
$$

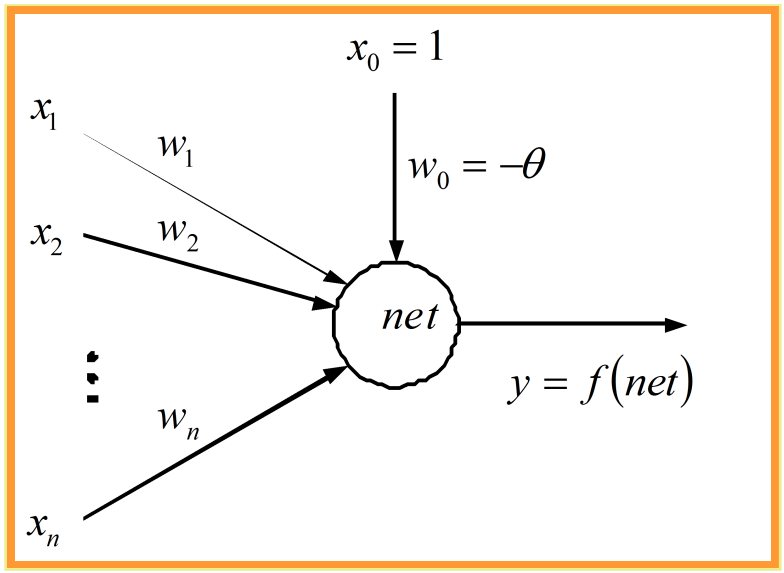


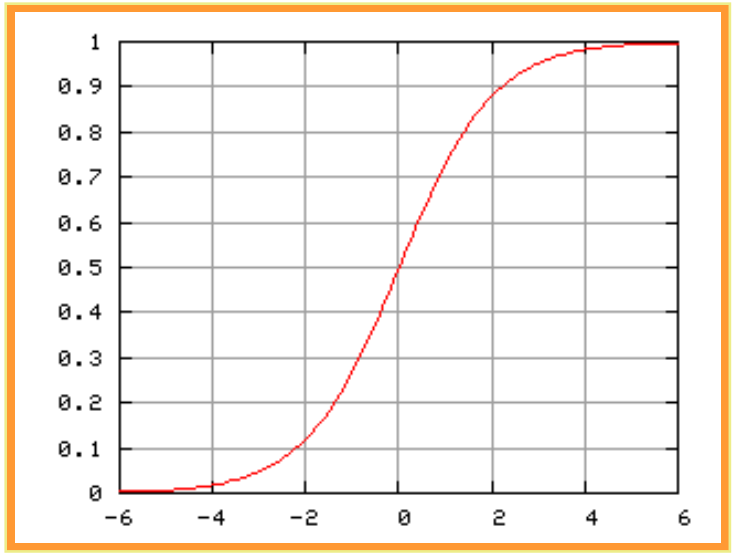
# [Matlab implementation]

```
%% Integration of Hodgkin--Huxley equations with Euler method
 clear; figure;%clf;
%% Setting parameters
 % Maximal conductances (in units of mS/cm^2); 1=K, 2=Na, 3=R
  g(1)=36; g(2)=120; g(3)=0.3;
 % Battery voltage ( in mV); 1=n, 2=m, 3=h
  E(1)=-12; E(2)=115; E(3)=10.613;
 % Initialization of some variables
  I_ext=0; V=-10; x=zeros(1,3); x(3)=1; t_rec=0;
 % Time step for integration
    dt=0.01;
%% Integration with Euler method
 for t=-30:dt:500
    if t==10; I_ext=6; end  % turns external current on at t=10
    if t==400; I_ext=0; end   % turns external current off at t=40
 % alpha functions used by Hodgkin-and Huxley
    Alpha(1)=(10-V)/(100*(exp((10-V)/10)-1));
    Alpha(2)=(25-V)/(10*(exp((25-V)/10)-1));
    Alpha(3)=0.07*exp(-V/20);
 % beta functions used by Hodgkin-and Huxley
    Beta(1)=0.125*exp(-V/80);
    Beta(2)=4*exp(-V/18);
    Beta(3)=1/(exp((30-V)/10)+1);
 % tau_x and x_0 (x=1,2,3) are defined with alpha and beta
    tau=1./(Alpha+Beta);
    x_0=Alpha.*tau;
 % leaky integration with Euler method
    x=(1-dt./tau).*x+dt./tau.*x_0;          % x is m,n,h
 % calculate actual conductances g with given n, m, h
    gnmh(1)=g(1)*x(1)^4;
    gnmh(2)=g(2)*x(2)^3*x(3);
    gnmh(3)=g(3);
 % Ohm's law
    I=gnmh.*(V-E);
 % update voltage of membrane
    V=V+dt*(I_ext-sum(I));
 % record some variables for plotting after equilibration
    if t>=0;
         t_rec=t_rec+1;
         x_plot(t_rec)=t;
         y_plot(t_rec)=V;
    end
```

# Definice neuronu

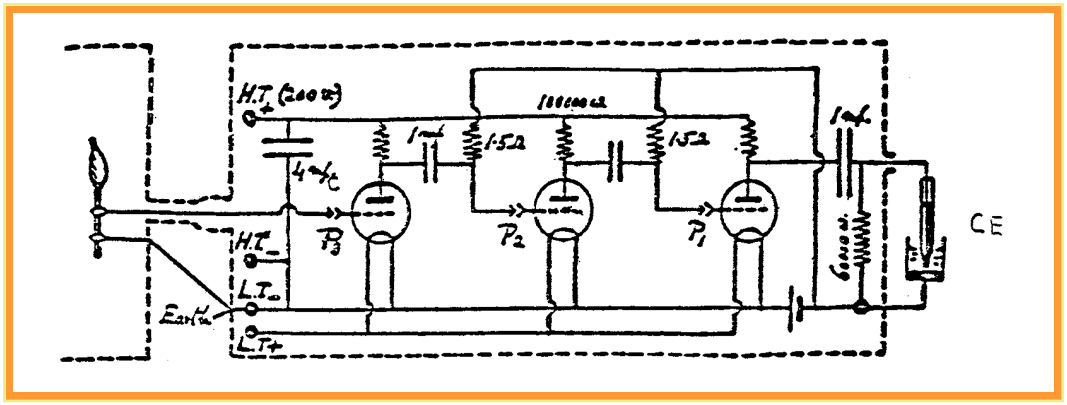
- Neuron je základní výpočetní jednotkou neuronových sítí
- Vstupy $x_i$ jsou váhovány parametry $\omega_i$
- $net = \sum_{i=1}^{n} x_i \omega_i + w_0 = \sum_{i=0}^{n} \vec{w}^t \vec{x}$
- Zavedení nelinearity do neuronové sítě $y = f(net)$, nejčastěji sigmoid třida, např, $\tanh$ či logistická funkce $y = f(net) = \frac{1}{1+\exp^{-\lambda * net}}$

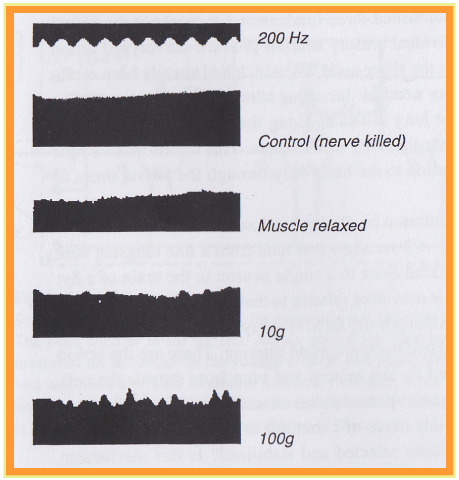


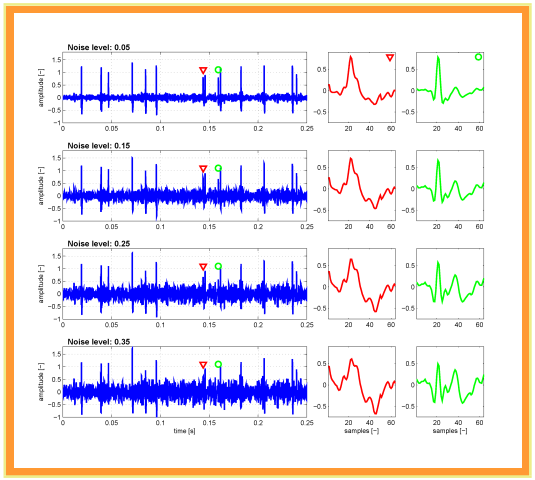



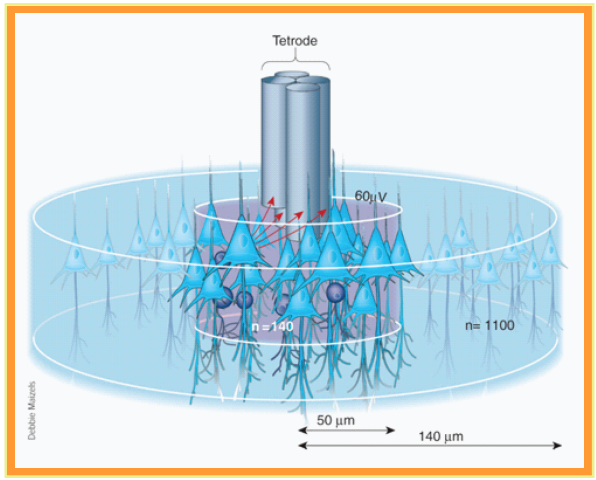
# Fyziologie

- Pionýrská práce barona Edgarda Adriana (Nobelova cena za medicínu v roce 1932)
  `http://nobelprize.org/nobel_prizes/medicine/laureates/1932/adrian-bio.html#`



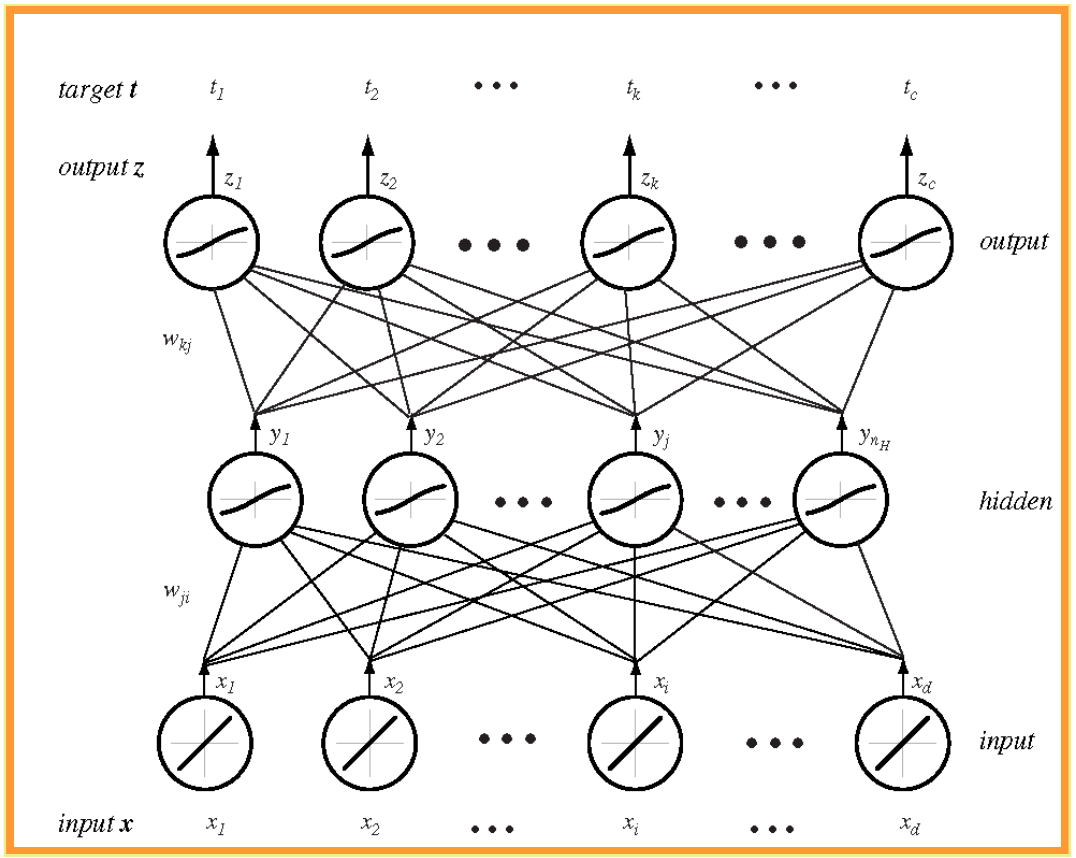


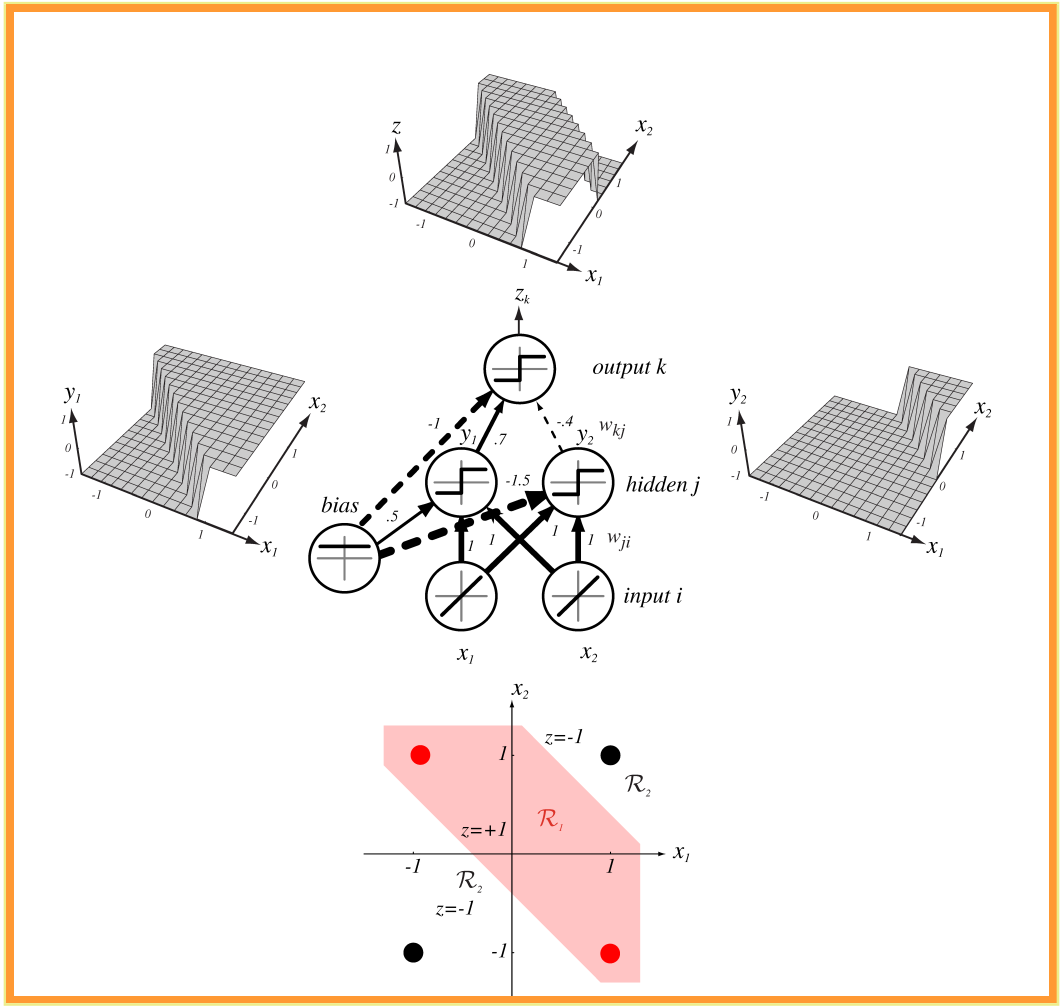
# Třívstvá neuronová síť ($d - n_H - c$)

- První vrstva je vstupní vrstva, akční funkce je zde lineární, počet neuronů se rovná dimenzi vstupního vektoru, $1 \ldots d$
- Druhá vrstva je skrytá, libovolný počet neuronů, $1 \ldots n_H$
- Třetí vrstva je výstupní vrstva, počet neuronů je nejčastěji roven počtu tříd, $1 \ldots c$



# Příklad - třívstvá neuronová sít´ - řešení XOR problému

- $0 \bigoplus 0 = 0,\ 1 \bigoplus 1 = 0,\ 1 \bigoplus 0 = 1,\ 0 \bigoplus 1 = 1$
- $-1 \bigoplus -1 = -1,\ 1 \bigoplus 1 = -1,\ 1 \bigoplus -1 = 1,\ -1 \bigoplus 1 = 1$



# Řešení XOR problému

- rozhodovací hranice skrytého neuronu $y_1$

$$x_1 + x_2 + 0,5 = 0 \begin{cases} \geq 0 & \text{if } y_1 = +1 \\ < 0 & \text{if } y_1 = -1 \end{cases}$$

- rozhodovací hranice skrytého neuronu $y_2$

$$x_1 + x_2 - 1.5 = 0 \begin{cases} \geq 0 & \text{if } y_2 = +1 \\ < 0 & \text{if } y_2 = -1 \end{cases}$$

- neuron ve výstupní vrstvě $z$

$$0.7y_1 - 0.4y_2 - 1 = 0 \begin{cases} \geq 0 & \text{if } z = +1 \\ < 0 & \text{if } z = -1 \end{cases}$$

# Aktivace neuronu

- aktivace $net_j$ skryté vrstvy $net_j = \sum_{i=1}^{d} x_i w_{ji} + w_{j0} = \sum_{i=0}^{d} x_i w_{ji} = \vec{w_j}^t \vec{x}$
- $i$ indexuje neuron ve vstupní vrstvě, $j$ skrytou vrstvu, $w_{ji}$ je váha neuron $j$ ve skryté vrstvě, který je spojen se vstupním neuronem $i$ (synapse).
- Výstup neuronu ve skryté vrstvě $y_j = f(net_j)$
- XOR problém

$$f(net) = sgn(net) \begin{cases} 1 & \text{net } \geq 0 \\ -1 & \text{net } < 0 \end{cases}$$

- funkce $f(.)$ se nazývá aktivační funkcí.
- Podobně, aktivační funkce $net_k$ neuronu ve výstupní vrstvě je dána jako $net_k = \sum_{j=1}^{n_H} y_j w_{kj} + w_{k0} = \sum_{j=0}^{n_H} y_j w_{kj} = \vec{w_k}^t \vec{y}$
- $k$ indexuje neuron ve výstupní vrstvě, $n_H$ je počet skrytých neuronů
- Výstup neuronu ve výstupní vrstvě $z_k = f(net_k)$
- V případě c tříd (výstupů), síť počítá $c$ diskriminačních funkcí $z_k = g_k(\vec{x})$ a klasifikuje vstup $\vec{x}$ podle největší diskriminační funkce $g_k(\vec{x}) \qquad \forall k = 1, \ldots c$

# Vybavování sítě - dopředná operace

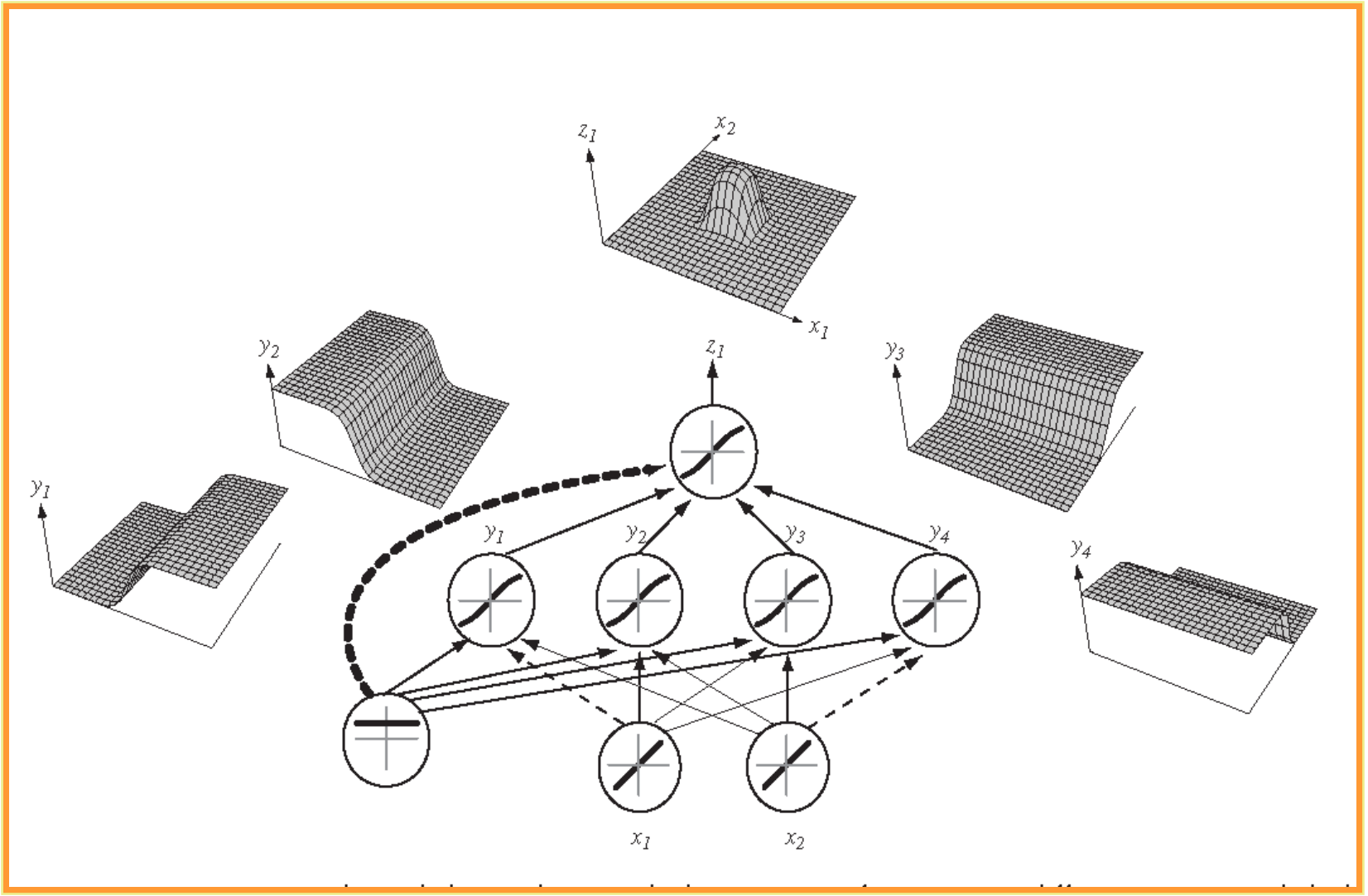
- Výstup sítě

$$g_k(\vec{x}) = z_k = f(\sum_{j=1}^{n_H} w_{kj} f(\sum_{i=1}^{d} w_{ji} x_i + w_{j0}) + w_{k0}) \qquad \forall k = 1 \dots c$$

- Skrytá vrstva umožnuje realizovat komplikované nelineární funkce
- Aktivační funkce - v každé vrstvě můžeme mít rozdílnou aktivační funkci, dokonce každý neuron může mít svoji vlastní aktivační funkci
- Pro zbytek přednášky předpokládáme, že máme jeden typ aktivační funkce
- **OTÁZKA: Může třívrstvá neuronová sít´ aproximovat jakoukoliv nelineární funkci?**
- **ODPOVĚD: ANO - díky A.Kolmogorovi**
  **Jákákoliv spojitá funkce může být implementovaná třívrstvou sítí za předpokladu dostatečného počtu $n_H$ skrytých neuronů,vhodných nelinearit a vah $w$.**

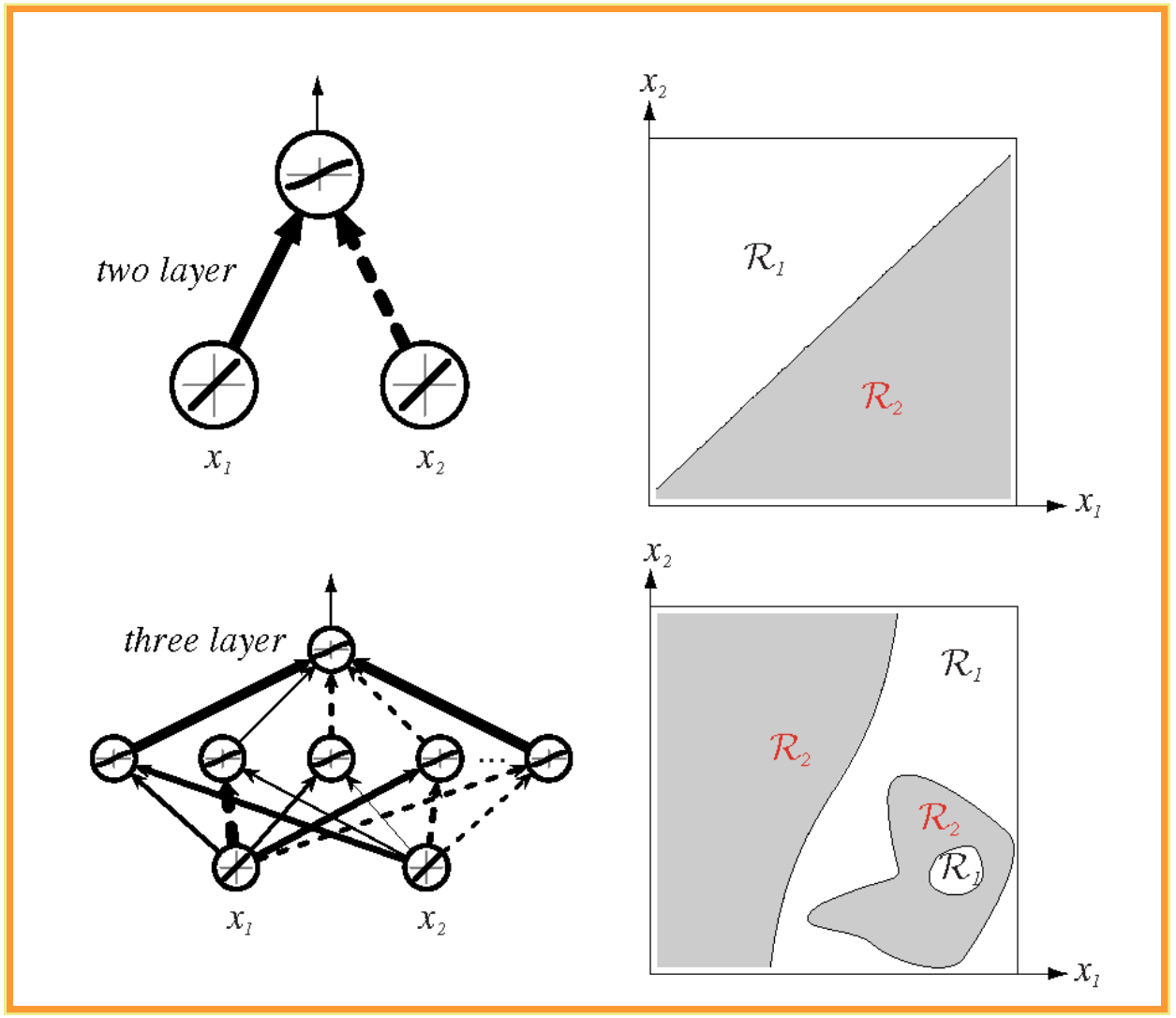
# Aproximace nelineární funkce

- Analogie s Fourierovým rozvojem!



# Příklad rozhodovací hranice

- Porovnání 2-vrstvé a 3-vrstvé sítě



# Andrej Kolmogorov

- Již na střední škole sestrojil "perpetuum mobile", jeho učitel nepřišel na použitý trik
- Studoval nejdříve historii na Moskevské státní univerzitě (nastoupi v 17 letech)
- První vědecký článek publikoval na téma vlastnictví nemovitostí v Novgorodu v období 15. a 16. století
- Největší přínos v teorii pravděpodobnosti a výpočetní složitosti

# Jak sít´naučíme ????

- Náš cíl je nastavit váhy na základě trénovacích dat a požadovaného výstupu $t_k$
- Odvodíme si metodu zpětného šíření chyby
- Necht´ $t_k$ je $k$ skutečný výstup a $z_k$ necht´je vypočtený výstup, kde $k = 1, \ldots, c$. Definujeme chybu jako

$$J(\vec{w}) = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2 = \frac{1}{2} \parallel \vec{t} - \vec{z} \parallel$$

- Algoritmus zpětného šíření chyby je založen na gradietním algorimu (viz percepton). Váhy jsou náhodně inicializovány a jsou měněny ve směru poklesu chyby

$$\triangle\vec{w} = -\eta\frac{\partial J}{\partial \vec{w}}$$

- $\eta$ je parametr ovlivnující rychlost učení - určuje relativní změnu vah

$$\vec{w}(m+1) = \vec{w}(m) + \triangle\vec{w}$$

- kde $m$ je m-tý použitý vzor $(\vec{x}_m, t_m)$

# Odvození

- Chyba pro váhy (skrytá-výstupní)

$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial z_k}\frac{\partial z_k}{\partial net_k}\frac{\partial net_k}{\partial w_{kj}} = \delta_k \frac{\partial net_k}{\partial w_{kj}}$$

- Platí $net_k = \sum_{j=0}^{n_H} y_j w_{kj} = \vec{w}_k{}^t \vec{y}$, tedy $\frac{\partial net_k}{\partial w_{kj}} = y_j$
- kde sensitivita $k$-tého neuronu je definována jako $\delta_k = -\frac{\partial J}{\partial net_k}$ a popisuje, jak se celková chyba mění s aktivací parametru $net_k$ neuronu, $\frac{\partial z_k}{\partial net_k} = f'(net_k)$

$$\delta_k = -\frac{\partial J}{\partial z_k}\frac{\partial z_k}{\partial net_k} = (t_k - z_k) f'(net_k)$$

- Váhy (skrytá-výstupní) se aktualizují jako

$$\boxed{\triangle w_{kj} = \eta \delta_k y_j = \eta (t_k - z_k) f'(net_k) y_j}$$

- Chyba pro váhy (vstupní-skrytá)

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}}$$

- Platí

$$\frac{\partial J}{\partial y_j} = \frac{\partial}{\partial y_j}\left[\frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2\right] = -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial y_j} =$$
$$-\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial net_k}\frac{\partial net_k}{\partial y_j} = -\sum_{k=1}^{c}(t_k - z_k)f^{'}(net_k)w_{kj}$$

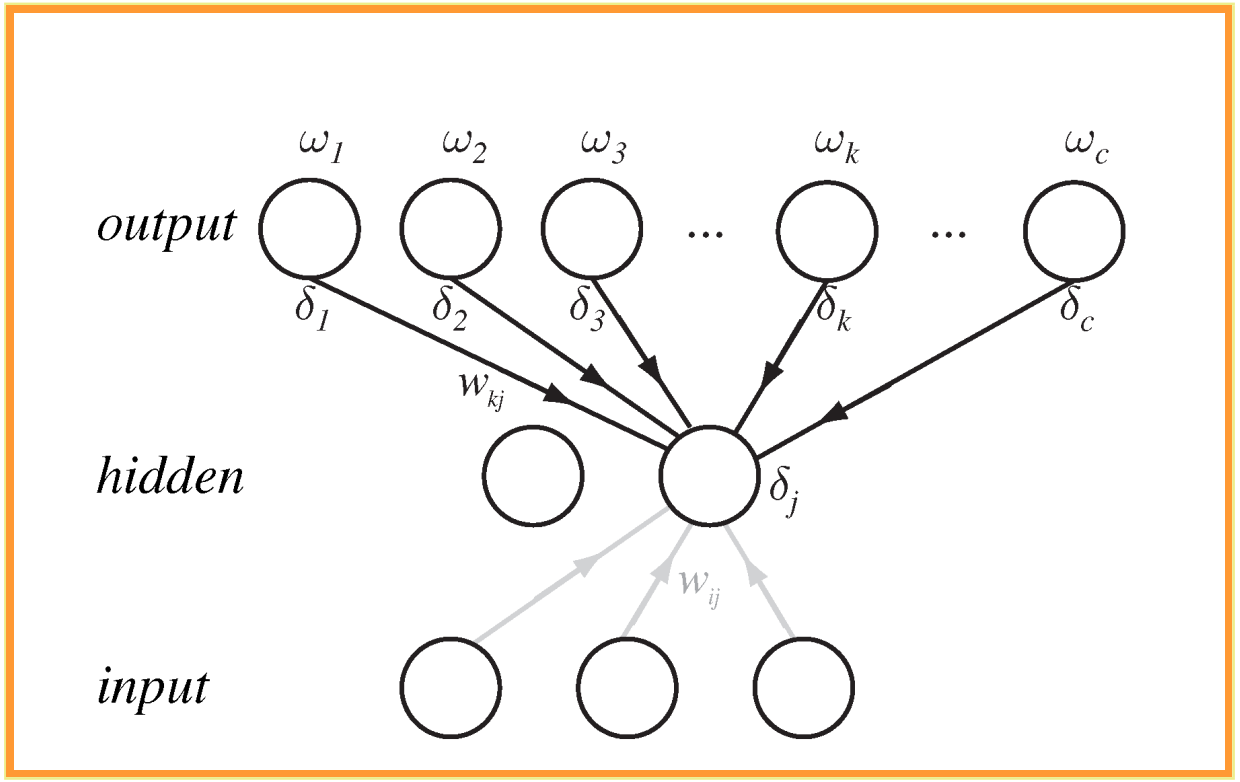
- Platí $\frac{\partial net_k}{\partial y_j} = w_{kj}$, protože $net_k = \sum_{j=0}^{n_H} y_j w_{kj} = \vec{w}_k{}^t \vec{y}$
- Takže $\frac{\partial J}{\partial y_j} = -\sum_{k=1}^{c}\delta_k w_{kj}$, protože $\delta_k = (t_k - z_k)f^{'}(net_k)$
- Zavedeme $\frac{\partial y_j}{\partial net_j} = f^{'}(net_j)$
- Zbývá vyčíslit $\frac{\partial net_j}{\partial w_{ji}} = x_i$, protože platí $net_j = \sum_{i=0}^{d} x_i w_{ji} = \vec{w}_j{}^t \vec{x}$
- Dosadíme a definujeme sensitivitu pro skrytou jednotku. Sensitivita je váhovaný součet výstupních sensitivit, znásoben aktivační funkcí skrytého neuronu

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}} = \underbrace{-\sum_{k=1}^{c}\delta_k w_{kj} f^{'}(net_j)}_{\delta_j} x_i$$

# Proč zpětné šíření chyby (back-propagation)?

- Pravidlo aktualizace vah (vstupní-skrytá) je

$$\triangle w_{ji} = \eta x_i \delta_j = \eta \sum (w_{kj}\delta_k) f'(net_j) x_i$$



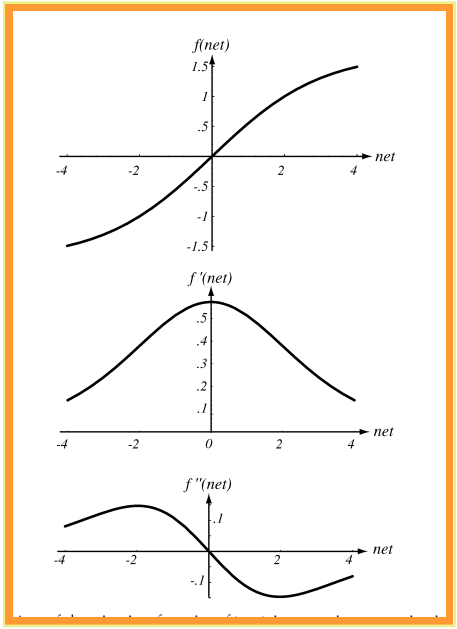
# Pseudo-kód

- pseudokód (inkrementální učení, stochastické)

```
1 begin initialize  network topology (# hidden units), w, criterion θ, η, m ← 0
2   do m ← m + 1
3       x^m ← randomly chosen pattern
4       w_ij ← w_ij + η δ_j x_i;   w_jk ← w_jk + η δ_k y_j
5   until ∇J(w) < θ
6 return w
7 end
```

– Implementace v Matlabu: *Backpropagation_Stochastic.m*. Použita funkce tanh, $a = 1.716$, $b = \frac{2}{3}$, aby $f'(0) \simeq 1$.

$$f(net) = a \tanh(b \star net) = \frac{2a}{1 + \exp^{b \star net}} - a$$



# Batch učení

- Mám $n$ vstupů $\vec{x_i}$ , vyjádřím celkovou chybu přes všechny vzorky jako

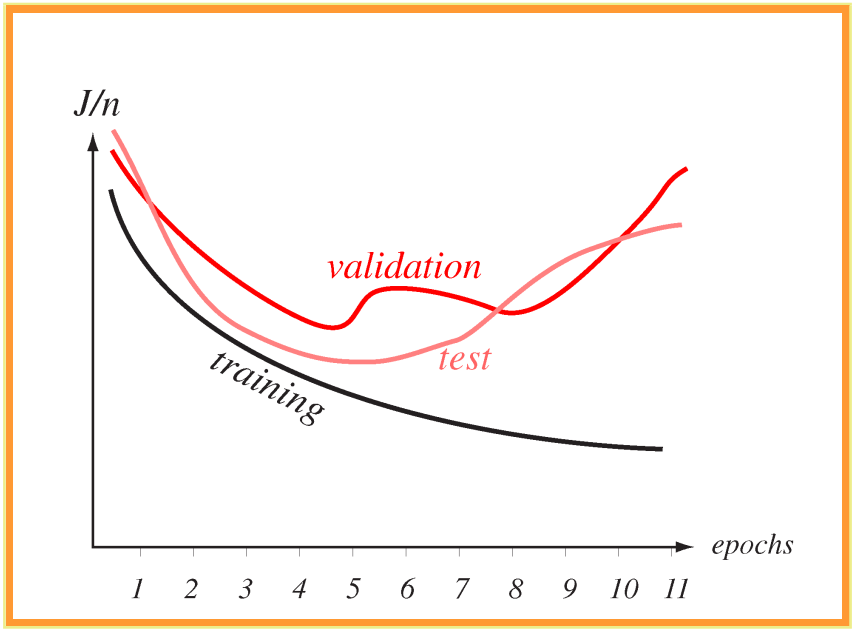
$$J = \sum_{p=1}^{n} J_p$$

- Není nutné vybírat vstupy postupně
- Epocha je jedna prezentace všech vstupů, krok 2: $r = r + 1$

```
1  begin initialize  network topology (# hidden units), w, criterion θ, η, r ← 0
2    do r ← r + 1 (increment epoch)
3       m ← 0; Δw_ij ← 0; Δw_jk ← 0
4       do m ← m + 1
5          x^m ← select pattern
6          Δw_ij ← Δw_ij + ηδ_j x_i;   Δw_jk ← Δw_jk + ηδ_k y_j
7       until m = n
8       w_ij ← w_ij + Δw_ij;   w_jk ← w_jk + Δw_jk
9    until ∇J(w) < θ
10 return w
11 end
```

# Validace

- Chyba trénovací množiny je monotonní klesající funkce díky gradietnímu algoritmu (klesám ve směru poklesu chyby)
- Rozdělení dat na trénovací,testovací a validační množinu. Validaci používám jako zastavovací kritérium (první minimum nebo celkové minimum pro konstantní počet epoch)




- DEMO - Neural Network Toolbox v Matlabu
  `http://www.mathworks.com/products/neuralnet/`
- Data pocházejí z UCI Machine Learning Repository
  `http://mlearn.ics.uci.edu/MLRepository.html`