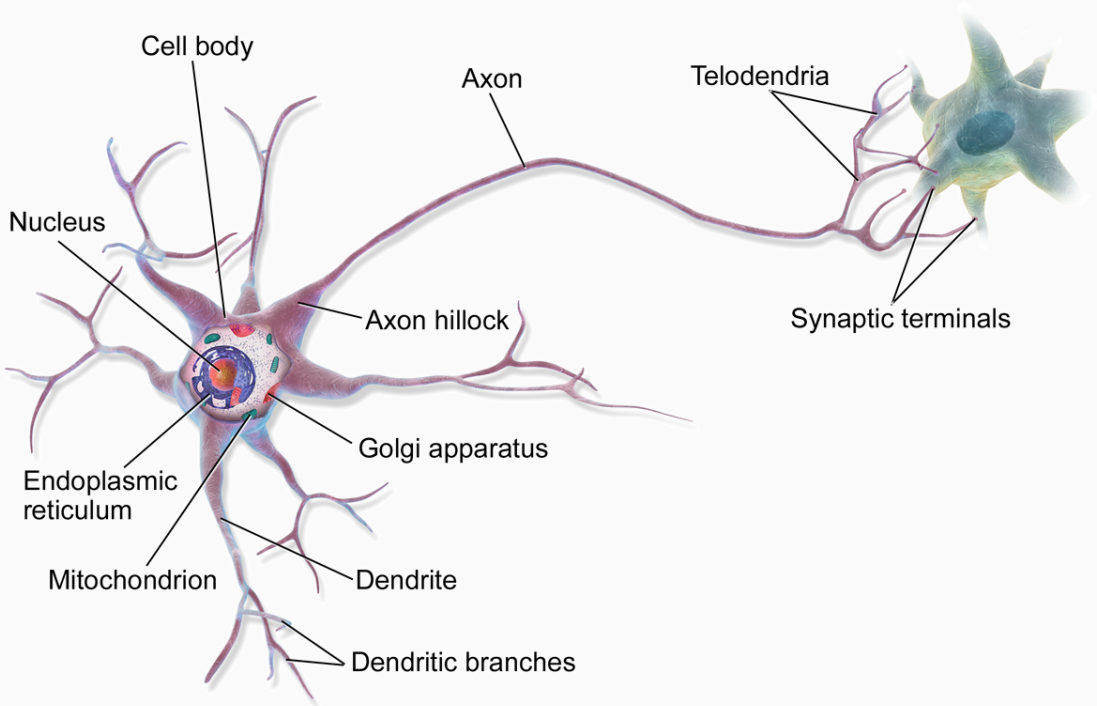
Egyrétegű előrecsatolt neuronhálók

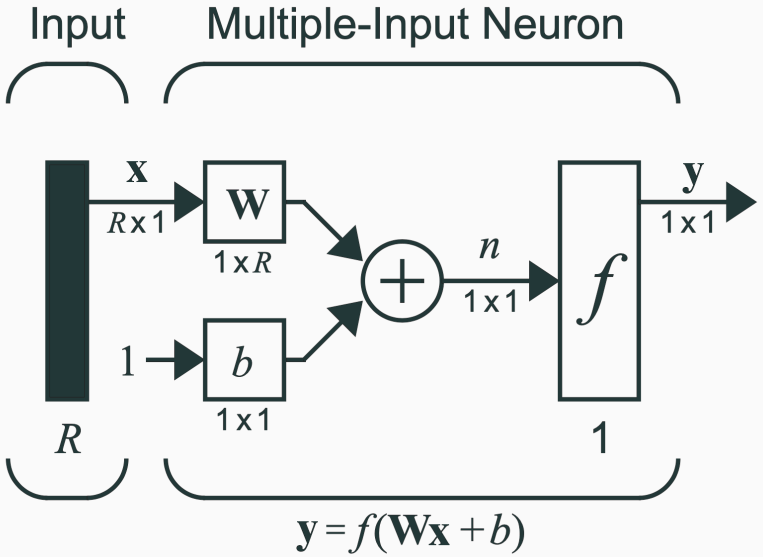
Neuron



- absztrakt neuron modell: $y = f\left(\sum_i (x_i w_i + b)\right)$
- x_i: a neuron bemenetei
- w_i: súlyzók
- b: bias
- f: aktivációs függvény
- y: a neuron kimenete

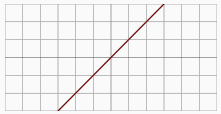
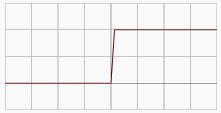
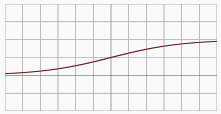
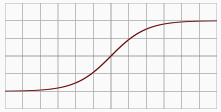
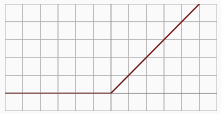
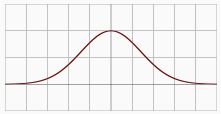
Egyrétegű előrecsatolt neuronháló

- egy kimenettel rendelkező egyrétegű előrecsatolt neuronháló:



- az egyszerűbb jelölésért: $x_0 = 1$ és $w_0 = b$
- NB: csak lineárisan szétválasztható problémákat tud megoldani

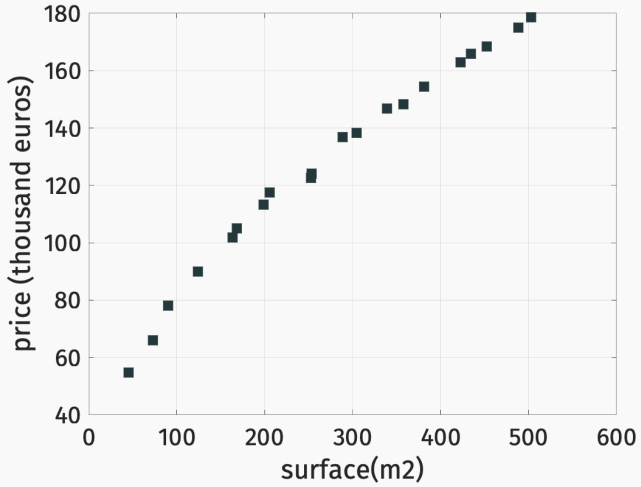
Aktivációs függvények

egység		$f(x) = x$	$f'(x) = 1$
lépcső		$f(x) = \begin{cases} 0, & \text{ha} \quad x < 0 \\ 1, & \text{ha} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0, & \text{ha} \quad x \neq 0 \\ ?, & \text{ha} \quad x = 0 \end{cases}$
sigmoid		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
tanh		$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$f'(x) = 1-f(x)^2$
relu		$f(x) = \begin{cases} 0, & \text{ha} \quad x \leq 0 \\ x, & \text{ha} \quad x > 0 \end{cases}$	$f'(x) = \begin{cases} 0, & \text{ha} \quad x \leq 0 \\ 1, & \text{ha} \quad x > 0 \end{cases}$
Gauss		$f(x) = e^{-x^2}$	$f'(x) = -2xf(x)$

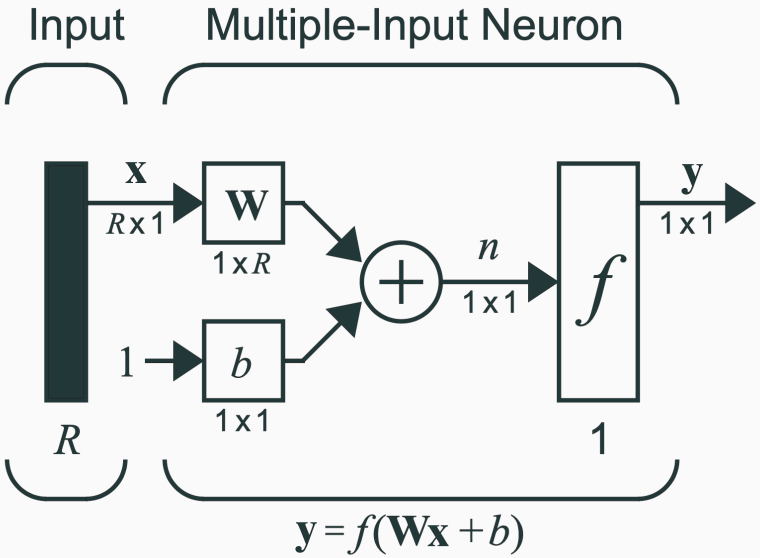
Felügyelt tanítás. Osztályozás vs regresszió

- felügyelt tanítás: felcímkézett adatok alapján (labelled data) próbáljuk modellezni azt az *f* függvényt, amely az adatokat generálta
- *fontos:* nem az adatpontok memorizálása a cél!
- osztályozás: a leképzés diszkrét, a kimenet valamilyen osztályt jelent
- regresszió: a leképzés folytonos
- példa:

sepal length	sepal width	petal length	petal width	class
5.1	3.5	1.4	0.2	iris-setosa
4.6	3.1	1.5	0.2	iris-setosa
7.0	3.2	4.7	1.4	iris-versicolor
6.4	3.2	4.5	1.5	iris-versicolor
5.7	2.8	4.5	1.3	iris-versicolor
6.3	3.3	6.0	2.5	iris-virginica
7.1	3.0	5.9	2.1	iris-virginica
6.3	2.9	5.6	1.8	iris-virginica
...				



Négyzetes hibafüggvény (quadratic loss function)



- az i. tanítási példához tartozó hiba

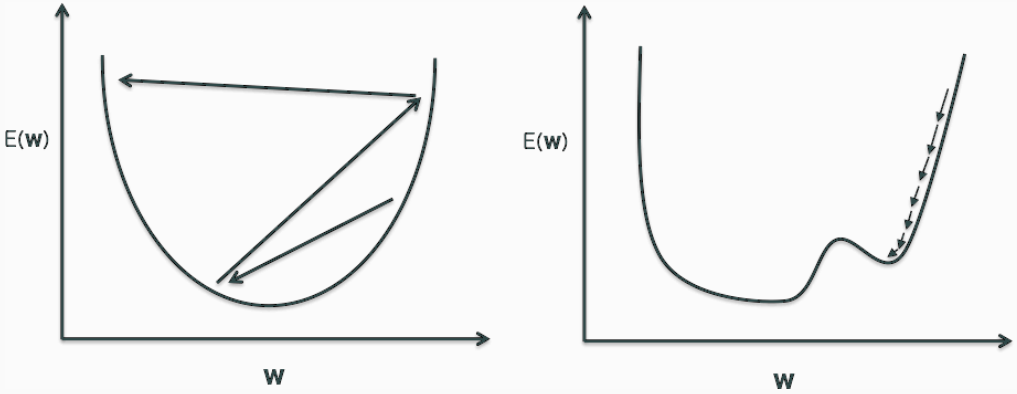
$$e_i = \frac{1}{2}(y_i - t_i)^2, \quad \text{ahol}$$

  - (x_i, t_i) a tanítási halmaz i. eleme
  - $y_i = f\left(\sum_k x_k^i w_k\right)$ a neurális háló kimenet az i. tanítási példára
- a (teljes) tanítási halmazhoz tartozó (globális) hiba

$$E = \sum_i e_i$$

Gradiens alapú tanítás

- tanítás, $\mathbf{w}$ alapú optimizáció: $\underset{\mathbf{w}}{\operatorname{argmin}} E(\mathbf{w})$
- a keresés iránya, gradiens: $\nabla E(\mathbf{w}) = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_n}\right]$
- gradiens keresés: $\Delta \mathbf{w} = -\eta \nabla E(\mathbf{w})$, ahol η a tanítási ráta



- adaptív tanítási ráta, pl RMSProp
  - minden w_i súlyzóra külön $v(w_i, t)$ együttható, amely az η-t skálázza
  - $\Delta w_i(t) = -\frac{\eta}{\sqrt{v(w_i,t)}} \nabla E(\mathbf{w})$
  - $v(w_i, t) = \gamma v(w_i, t-1) + (1-\gamma)\left(\nabla E(\mathbf{w})\right)^2$, ahol γ a felejtési faktor
- momentum módszer: $\Delta \mathbf{w}(t+1) = -\eta \nabla E(\mathbf{w}(t)) + \varepsilon \Delta \mathbf{w}(t)$
- adaptív momentum (ADAM)
- másodrendű módszerek (second order methods)

Delta szabály

- Delta szabály: $\Delta w_k = -\eta \frac{\partial E}{\partial w_k}$
- $\frac{\partial E}{\partial w_k} = \sum_i \frac{\partial e_i}{\partial w_k}$
- négyzetes hibafüggvény estetén:

$$\frac{\partial e_i}{\partial w_k} = \frac{\partial \left(\frac{1}{2} (y_i - t_i)^2 \right)}{\partial w_k} \tag{1}$$

$$= (y_i - t_i) \frac{\partial y_i}{\partial w_k} \tag{2}$$

$$= (y_i - t_i) \frac{\partial f(\mathbf{x}^i \mathbf{w})}{\partial w_k} \tag{3}$$

$$= (y_i - t_i) f'(\mathbf{x}^i \mathbf{w}) \frac{\partial \left(\sum_k x_k^i w_k \right)}{\partial w_k} \tag{4}$$

$$= (y_i - t_i) f'(\mathbf{x}^i \mathbf{w}) x_k^i \tag{5}$$

Batch/offline tanítás

- a teljes tanítási halmaz bejárása után frissítjük a súlyzókat

```
function w = OfflineLearning(x, d, f, gradf, lr, stop)
[~, n] = size(x);
w = randn(n,size(d,2));
epoch = 0;
while true
    v = x * w;
    y = f(v);
    e = y - d;                  % using square loss function
    g = x' * (e .* gradf(v));   %
    w = w - lr * g;
    E = sum(e(:).^2);
    if stop(E, epoch), break; end
    epoch = epoch + 1;
end
```

demo

Online tanítás

- minden tanítási példa után frissítjük a súlyzókat

```
function w = OnlineLearning(x, d, f, gradf, lr, stop)
[N, n] = size(x);
w = randn(n,size(d,2));
epoch = 0;
while true
    E = 0;
    for i = randperm(N)
        vi = x(i,:)*w;
        yi = f(vi);
        ei = yi-d(i,:);                  % using square loss function
        gi = x(i,:)' * ei .* gradf(vi);  %
        w = w - lr * gi;
        E = E + sum(ei.^2);
    end
    if stop(E, epoch), break; end
    epoch = epoch + 1;
end
```

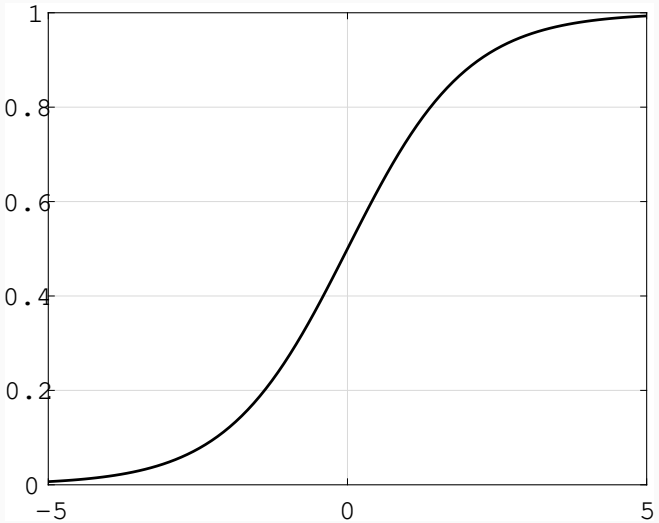
Gyakorlati megfontolások

- online vs offline/batch learning, mini batches
- a tanítási adatok előfeldolgozása: skálázás, class-imbalance, nominális adatok kezelése
- a `w` kezdeti értékei: `rand`, `randn`, nem felügyelt tanítás (RBM)
- tanítási ráta: rögzített, adaptív
- aktivációs függvény:
  - szimmetria
  - szaturálódás
  - eltűnő gradiens (vanishing gradient)
    - a bináris cél értékek (target output) újrakódolása (pl -1 -> -0.8, 1 -> 0.8)
- megállási feltétel
- tanítás, *validálás*, tesztelés

Osztályozás logisztikai regresszióval

- példa: tumor mérete, formája, ... stb input, 1-es osztályozás: malignant, 0-s osztályozás benign
- értelmezés: $p(y=1|x;w)$
- aktivációs függvény, sigmoid: $f(z)=\frac{1}{1+e^{-z}}$
- neuron kimenete: $y=\frac{1}{1+e^{-\mathbf{wx}}}$
- konvex hibafüggvény:

$$e(d,y)=\begin{cases} -\ln(y), & \text{ha} \quad d=1 \\ -\ln(1-y), & \text{ha} \quad d=0 \end{cases}$$



- több kimenetre softmax: $f(z_i)=\frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$

Könyvtárak

- scikit-learn, keras, pytorch, caffee, tensorflow (python)
- neural network toolbox (matlab)

Könyvészet, ábrák

- Christopher Bishop: Pattern Recognition and Machine Learning, 2006
- Simon Haykin: Neural Networks and Learning Machines, 2009
- Martin Hagan: Neural Network Design 2nd ed, 2014
- Andrew Ng: Machine Learning (Stanford/Coursera)
- Wikipedia