zumz5

June 1, 2018

0.1 Uczenie maszynowe UMZ 2017/2018

1 5. Sieci neuronowe

In [13]:
```
%matplotlib inline

import numpy as np
```

1.1 5.1. Perceptron

In [14]:
```
from IPython.display import YouTubeVideo
YouTubeVideo('cNxadbrN_aI', width=800, height=600)
```

Out[14]:

1.1.1 Pierwszy perceptron liniowy

- Frank Rosenblatt, 1957
- aparat fotograficzny podczony do 400 fotokomórek (rozdzielczo obrazu: 20 x 20)
- wagi – potencjometry aktualizowane za pomoc silniczków

1.1.2 Uczenie perceptronu

Cykl uczenia perceptronu Rosenblatta:

1. Sfotografuj plansz z kolejnym obiektem.
2. Zaobserwuj, która lampka zapalia si na wyjciu.
3. Sprawd, czy to jest waciwa lampka.
4. Wylij sygna nagrody" lub kary".

1.1.3 Funkcja aktywacji

Funkcja bipolarna:

$$g(z) = \begin{cases} 1 & \text{gdy } z > \theta_0 \\ -1 & \text{wpp.} \end{cases}$$

gdzie $z = \theta_0 x_0 + \ldots + \theta_n x_n$, θ_0 to próg aktywacji, $x_0 = 1$.

```
In [15]: def bipolar_plot():
             matplotlib.rcParams.update({'font.size': 16})

             plt.figure(figsize=(8,5))
             x = [-1,-.23,1]
             y = [-1, -1, 1]
             plt.ylim(-1.2,1.2)
             plt.xlim(-1.2,1.2)
             plt.plot([-2,2],[1,1], color='black', ls="dashed")
             plt.plot([-2,2],[-1,-1], color='black', ls="dashed")
             plt.step(x, y, lw=3)
             ax = plt.gca()
             ax.spines['right'].set_color('none')
             ax.spines['top'].set_color('none')
             ax.xaxis.set_ticks_position('bottom')
             ax.spines['bottom'].set_position(('data',0))
             ax.yaxis.set_ticks_position('left')
             ax.spines['left'].set_position(('data',0))

             plt.annotate(r'$\theta_0$',
                          xy=(-.23,0), xycoords='data',
                          xytext=(-50, +50), textcoords='offset points', fontsize=26,
```

```
                        arrowprops=dict(arrowstyle="->"))

            plt.show()
```

```
In [16]: bipolar_plot()
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-16-cf1bc21fbe41> in <module>()
----> 1 bipolar_plot()


<ipython-input-15-40cd09f35fc3> in bipolar_plot()
      1 def bipolar_plot():
----> 2     matplotlib.rcParams.update({'font.size': 16})
      3
      4     plt.figure(figsize=(8,5))
      5     x = [-1,-.23,1]


NameError: global name 'matplotlib' is not defined
```

1.1.4 Perceptron – schemat

1.1.5 Perceptron – zasada dziaania

1. Ustal wartoci pocztkowe θ (wektor 0 lub liczby losowe blisko 0).
2. Dla kadego przykadu $(x^{(i)}, y^{(i)})$, dla $i = 1, \ldots, m$
   - Oblicz warto wyjcia $o^{(i)}$:

$$o^{(i)} = g(\theta^T x^{(i)}) = g(\sum_{j=0}^{n} \theta_j x_j^{(i)})$$

   - Wykonaj aktualizacj wag (tzw. *perceptron rule*):

$$\theta := \theta + \Delta\theta$$

$$\Delta\theta = \alpha(y^{(i)} - o^{(i)})x^{(i)}$$

$$\theta_j := \theta_j + \Delta\theta_j$$

Jeeli przykad zosta sklasyfikowany **poprawnie**:

- $y^{(i)} = 1$ oraz $o^{(i)} = 1$:

$$\Delta\theta_j = \alpha(1-1)x_j^{(i)} = 0$$

- $y^{(i)} = -1$ oraz $o^{(i)} = -1$:

$$\Delta\theta_j = \alpha(-1 - -1)x_j^{(i)} = 0$$

Czyli: jeeli trafie, to nic nie zmieniaj.

$$\theta_j := \theta_j + \Delta\theta_j$$

Jeeli przykad zosta sklasyfikowany **niepoprawnie**:

- $y^{(i)} = 1$ oraz $o^{(i)} = -1$:

$$\Delta\theta_j = \alpha(1 - -1)x_j^{(i)} = 2\alpha x_j^{(i)}$$

- $y^{(i)} = -1$ oraz $o^{(i)} = 1$:

$$\Delta\theta_j = \alpha(-1 - 1)x_j^{(i)} = -2\alpha x_j^{(i)}$$

Czyli: przesu wagi w odpowiedni stron.

1.1.6 Perceptron – zalety i wady

Zalety: * intuicyjny i prosty * atwy w implementacji * jeeli dane mona liniowo oddzieli, algorytm jest zbieny w skoczonym czasie

Wady: * jeeli danych nie mona oddzieli liniowo, algorytm nie jest zbieny

```
In [ ]: def plot_perceptron():
            plt.figure(figsize=(12,3))

            plt.subplot(131)
            plt.ylim(-0.2,1.2)
            plt.xlim(-0.2,1.2)

            plt.title('AND')
            plt.plot([1,0,0], [0,1,0], 'ro', markersize=10)
            plt.plot([1], [1], 'go', markersize=10)

            ax = plt.gca()
            ax.spines['right'].set_color('none')
            ax.spines['top'].set_color('none')
            ax.xaxis.set_ticks_position('none')
            ax.spines['bottom'].set_position(('data',0))
            ax.yaxis.set_ticks_position('none')
            ax.spines['left'].set_position(('data',0))

            plt.xticks(np.arange(0, 2, 1.0))
            plt.yticks(np.arange(0, 2, 1.0))


            plt.subplot(132)
            plt.ylim(-0.2,1.2)
            plt.xlim(-0.2,1.2)
```

```
            plt.plot([1,0,1], [0,1,1], 'go', markersize=10)
            plt.plot([0], [0], 'ro', markersize=10)

            ax = plt.gca()
            ax.spines['right'].set_color('none')
            ax.spines['top'].set_color('none')
            ax.xaxis.set_ticks_position('none')
            ax.spines['bottom'].set_position(('data',0))
            ax.yaxis.set_ticks_position('none')
            ax.spines['left'].set_position(('data',0))

            plt.title('OR')
            plt.xticks(np.arange(0, 2, 1.0))
            plt.yticks(np.arange(0, 2, 1.0))


            plt.subplot(133)
            plt.ylim(-0.2,1.2)
            plt.xlim(-0.2,1.2)

            plt.title('XOR')
            plt.plot([1,0], [0,1], 'go', markersize=10)
            plt.plot([0,1], [0,1], 'ro', markersize=10)

            ax = plt.gca()
            ax.spines['right'].set_color('none')
            ax.spines['top'].set_color('none')
            ax.xaxis.set_ticks_position('none')
            ax.spines['bottom'].set_position(('data',0))
            ax.yaxis.set_ticks_position('none')
            ax.spines['left'].set_position(('data',0))

            plt.xticks(np.arange(0, 2, 1.0))
            plt.yticks(np.arange(0, 2, 1.0))

            plt.show()
```

```
In [ ]: plot_perceptron()
```

1.1.7 Funkcje aktywacji

Zamiast funkcji bipolarnej moemy zastosowa funkcj sigmoidaln jako funkcj aktywacji.

```
In [ ]: def plot_activation_functions():
            plt.figure(figsize=(16,7))
            plt.subplot(121)
            x = [-2,-.23,2]
```

```
    y = [-1, -1, 1]
    plt.ylim(-1.2,1.2)
    plt.xlim(-2.2,2.2)
    plt.plot([-2,2],[1,1], color='black', ls="dashed")
    plt.plot([-2,2],[-1,-1], color='black', ls="dashed")
    plt.step(x, y, lw=3)
    ax = plt.gca()
    ax.spines['right'].set_color('none')
    ax.spines['top'].set_color('none')
    ax.xaxis.set_ticks_position('bottom')
    ax.spines['bottom'].set_position(('data',0))
    ax.yaxis.set_ticks_position('left')
    ax.spines['left'].set_position(('data',0))

    plt.annotate(r'$\theta_0$',
                 xy=(-.23,0), xycoords='data',
                 xytext=(-50, +50), textcoords='offset points', fontsize=26,
                 arrowprops=dict(arrowstyle="->"))

    plt.subplot(122)
    x2 = np.linspace(-2,2,100)
    y2 = np.tanh(x2+ 0.23)
    plt.ylim(-1.2,1.2)
    plt.xlim(-2.2,2.2)
    plt.plot([-2,2],[1,1], color='black', ls="dashed")
    plt.plot([-2,2],[-1,-1], color='black', ls="dashed")
    plt.plot(x2, y2, lw=3)
    ax = plt.gca()
    ax.spines['right'].set_color('none')
    ax.spines['top'].set_color('none')
    ax.xaxis.set_ticks_position('bottom')
    ax.spines['bottom'].set_position(('data',0))
    ax.yaxis.set_ticks_position('left')
    ax.spines['left'].set_position(('data',0))

    plt.annotate(r'$\theta_0$',
                 xy=(-.23,0), xycoords='data',
                 xytext=(-50, +50), textcoords='offset points', fontsize=26,
                 arrowprops=dict(arrowstyle="->"))

    plt.show()
```

In [ ]:
```
plot_activation_functions()
```