KMI/ZZD – Získávání znalostí z dat

Shlukovnání

Jan Konečný

6. března 2012

Měření podobnosti a vzdálenosti

Výsledek shlukování ve velké míře závisí na měření podobnosti a vzdálenosti.

Vybrané funkce vzdálenosti

1. Euklidovská vzdálenost

$$\mathrm{d}(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

2. Hammingova vzdálenost

$$\mathrm{d}(\vec{x}, \vec{y}) = \sum_{i=1}^{n} |x_i - y_i|$$

3. Tchebyschevova vzdálenost

$$\mathrm{d}(\vec{x}, \vec{y}) = \max_{i=1}^{n} |x_i - y_i|$$

4. Minkowského vzdálenost

$$\mathrm{d}(\vec{x},\vec{y}) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}, \quad p > 0$$

5. Canberrova vzdálenost

$$\mathrm{d}(\vec{x},\vec{y}) = \sum_{i=1}^{n} \frac{|x_i - y_i|}{x_i + y_i}, \quad x_i \text{ a } y_i \text{ jsou kladné}$$

6. Úhlová separace

$$\mathrm{d}(\vec{x},\vec{y}) = \frac{\sum_{i=1}^{n} x_i y_i}{\left[\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2\right]^{\frac{1}{2}}}$$

Vzdálenost a podobonost binárních dat

Uvažujme dva binární vektory

$$\vec{x} = [x_1, x_2, \ldots, x_n]^T \quad \text{a} \quad \vec{y} = [y_1, y_2, \ldots, y_n]^T.$$

Porovnáme je po složkách, počítáme specifické kombinace 0 a 1

$$\begin{aligned} a &: x_k = 1 \text{ a } y_k = 1 \\ b &: x_k = 0 \text{ a } y_k = 1 \\ c &: x_k = 1 \text{ a } y_k = 0 \\ d &: x_k = 0 \text{ a } y_k = 0 \end{aligned}$$

	1	0
1	a	b
0	c	d

Následně se zjistí podobonost (viz následující slajdu)

- koeficient shody $\dfrac{a+b}{a+b+c+d}$
- Russell & Rao $\dfrac{a}{a+b+c+d}$
- Jacard index $\dfrac{a}{a+b+c}$
- Czekanowski index $\dfrac{2a}{2a+b+c}$

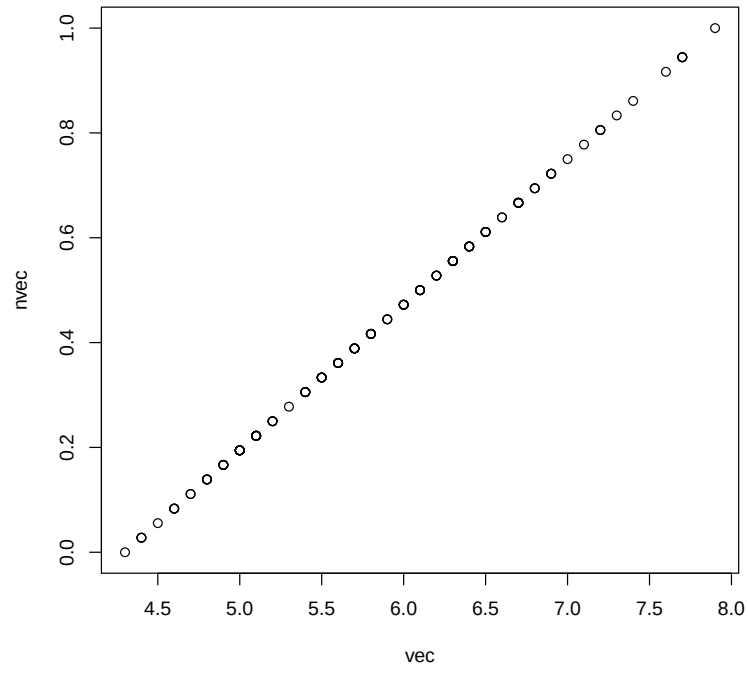
Problém: každý atribut reprezentuje něco jiného.
Co udělá větší vzdálenost: rozdíl 5kg v atributu *váha* nebo rozdíl 5cm v atributu *výška*.

- konstruovat metriku s doménovým expertem
- provést normalizaci – namapovat data na jeden interval, třeba $[0, 1]$

Lineární transformace

$$v_{i,\text{lin.norm}} = \frac{v_i - \min(v_1, \ldots, v_n)}{\max(v_1, \ldots, v_n) - \min(v_1, \ldots, v_n)}$$

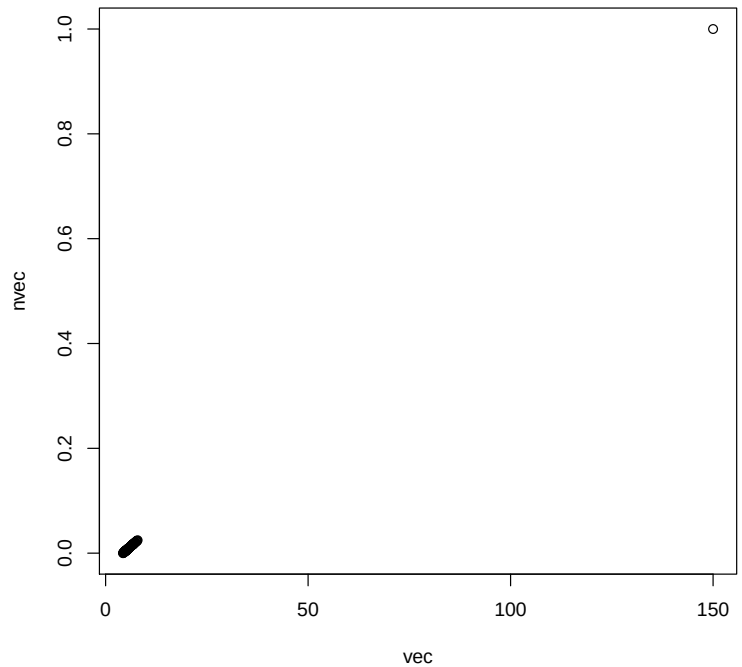
```
> vec <- iris[,1]
> max_v <- max(vec)
> min_v <- min(vec)
> min_v
[1] 4.3
> max_v
[1] 7.9
> nvec <- vapply(vec,0, FUN=function(vi)
                  {(vi - min_v)/(max_v-min_v)})
```



Problém: co když pak přijde hodnota, která je mimo interval min–max

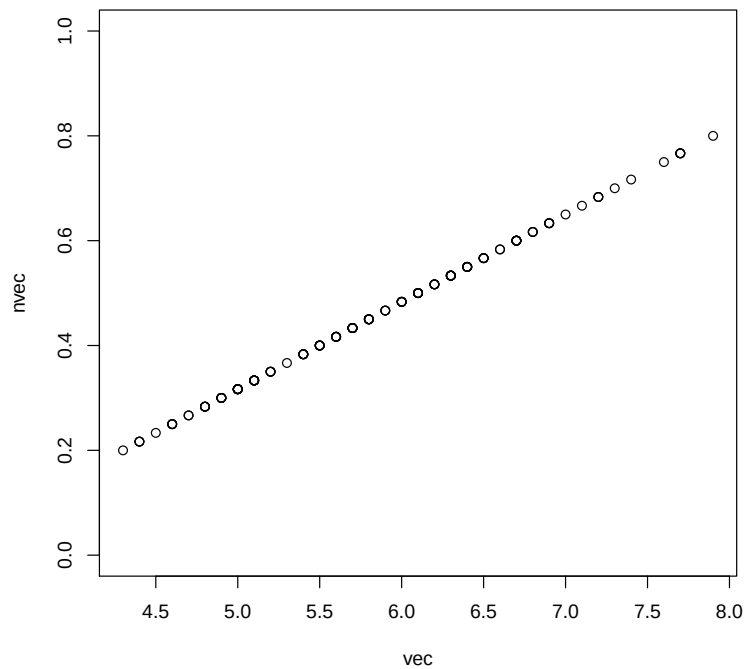
Jiný problém: co když je v datech odlehlý bod?

```
> vec[151]<-150
> nvec <- vapply(vec,0, FUN=function(vi)
                 {(vi - min_v)/(max_v-min_v)})
> plot(vec,nvec)
```



Co s tím: nemapujeme na [0,1] ale na [a,1-a], a hodnoty mimo „vtlačíme“ do $[0, a)$ a $(1 - a, 0]$.

```
> nvec <- vapply(vec,0, FUN=function(vi)
      {(vi - min_v)/(max_v-min_v)*(1-2*a)+a})
> plot(vec,nvec,ylim=0:1)
```

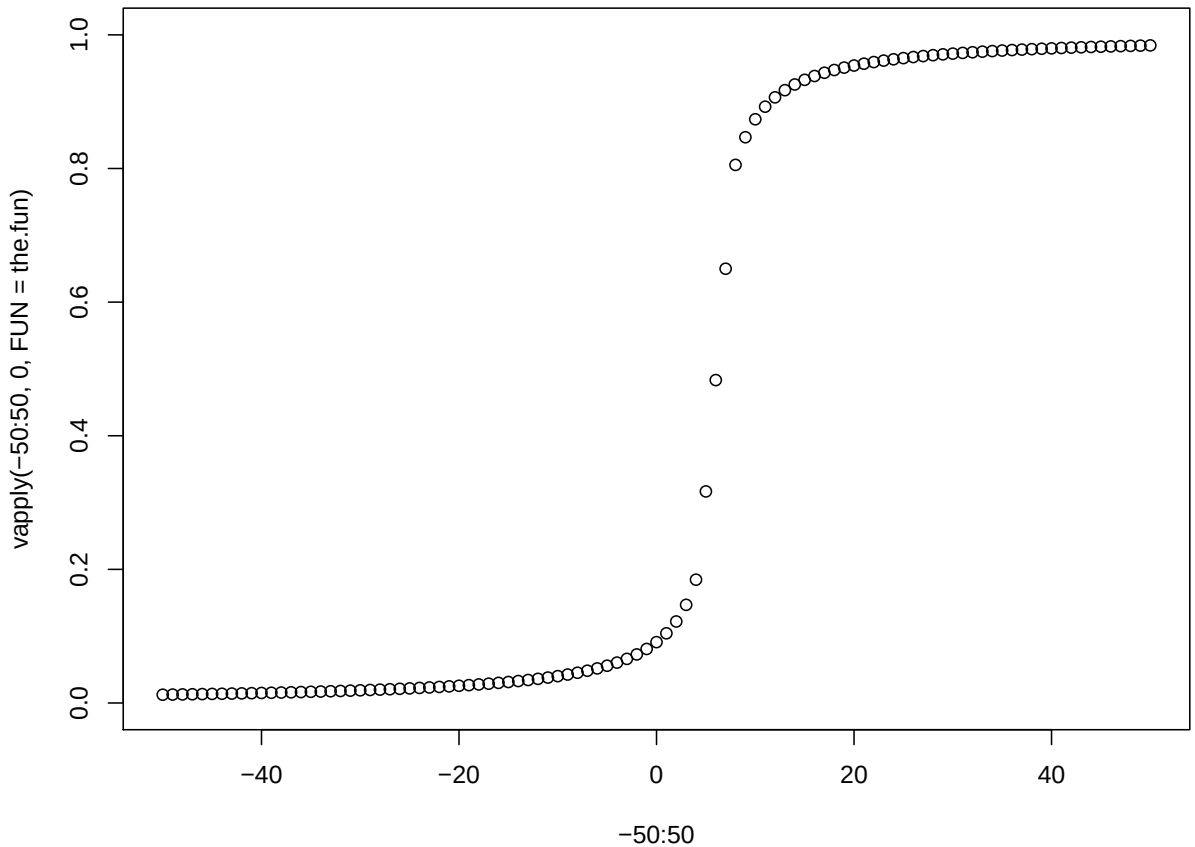


Jak teda na to vtlačení:

$$v_{i,\text{norm}} = \begin{cases} 1 - a/v_{i,\text{lin.norm}} & \text{pokud } v_i > u \\ a/(1 - v_{i,\text{lin.norm}}) & \text{pokud } v_i < l \\ v_{i,\text{lin.norm}} \cdot (1 - 2a) + a & \text{jinak} \end{cases}$$

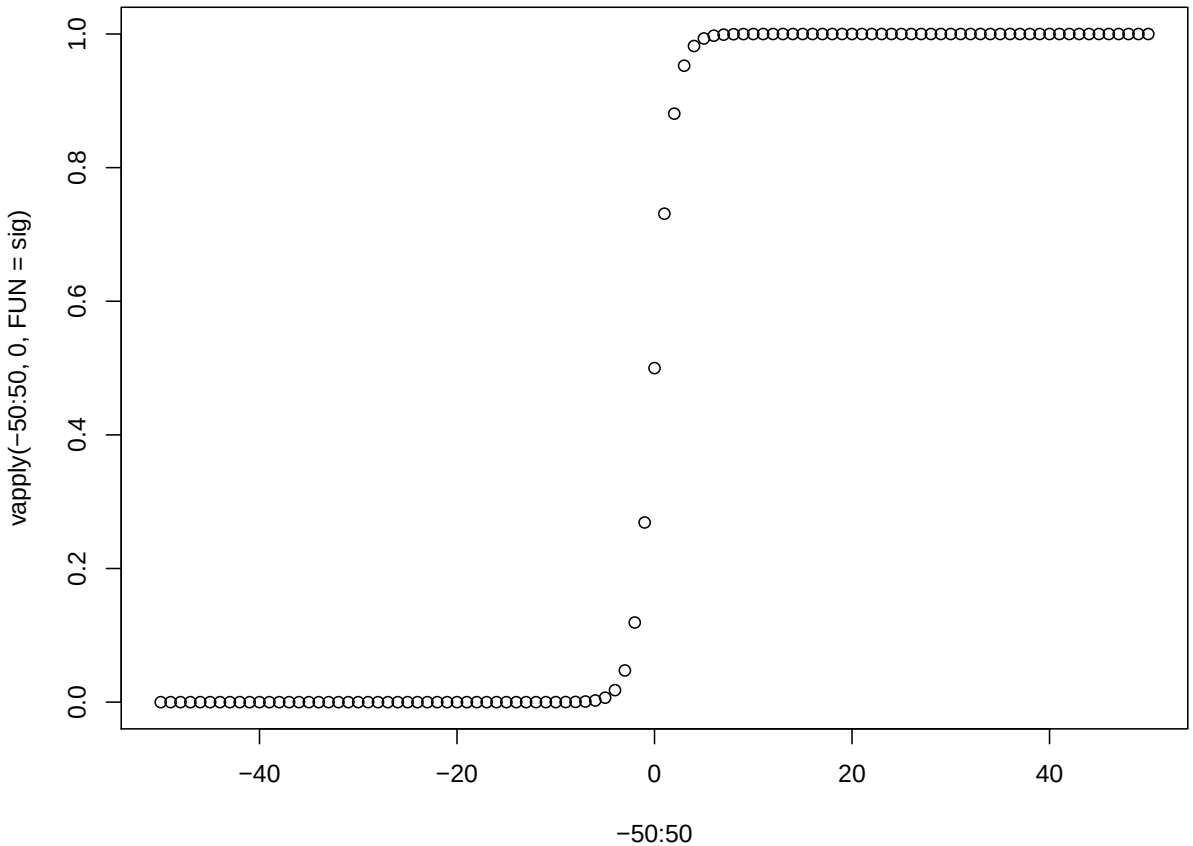
```
the.lin.fun <- function(vi)
{
 (vi - min_v)/(max_v-min_v)
}

the.fun <- function(vi)
{
 if (vi > max_v) return (1 - a * (1/the.lin.fun(vi)))
 if (vi < min_v) return (     a * (1/(1 - the.lin.fun(vi
 return(the.lin.fun(vi)*(1-2*a)+a)
}
```

```
> plot(-50:50,vapply(-50:50,0, FUN=the.fun),ylim=0:1)
```



To se celkem podobá simoidě

```
> sig <- function(x) {1 / (1 + exp(-x))}
> plot(-50:50, vapply(-50:50, 0, FUN=sig))
```



Sigmoida se taky dá použít pro normalizaci, ale předtím je potřeba data prohnat následujícím

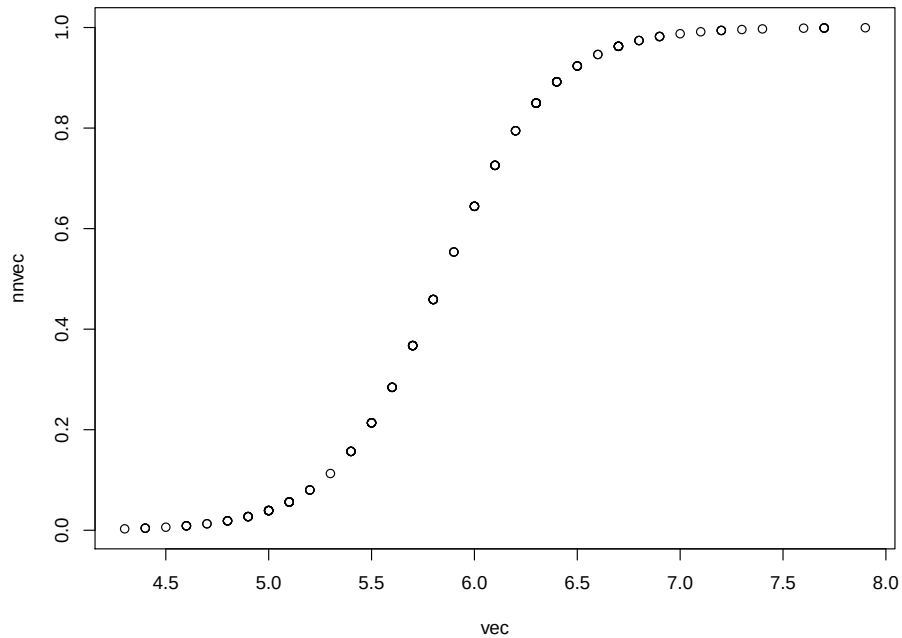
$$v_t = \frac{v_i - \mu}{\lambda \cdot \sigma/(2\pi)},$$

kde

- μ je průměr
- σ je směrodatná odchylka
- λ je parametr označující, kolik směrodatných odchylek má dávat lineární odpověď (obvykle se používá $\lambda = 2$)

```
> nvec  <- vapply( vec, 0, FUN=function(x)
            {(x - mean_v)/(2 * (sd_v / 2) * pi)})
> nnvec <- vapply(nvec, 0, FUN=sig)
```

```
> plot(vec,nnvec)
```



To je pak známo jako *softmax*