Kapitola 13 - Shlukování dat sítí bez učitele

Demonstrace použití neuronové sítě bez učitele k hledání shluků v datech.

Načtení knihovny NeuralNetworks

Nejdříve načteme knihovnu neuronových sítí.

In[3]:=
```
<< NeuralNetworks`
```

Pokud pracujete v Mathematice 8.0, vypněte ještě zobrazování chybové hlášky Remove::rmnsm. Tuto hlášku vyhazují funkce knihovny NeuralNetworks. Na funkci knihovny toto nemá žádný vliv.

In[4]:=
```
Off[Remove::rmnsm]
```

Vytvoření trénovacích dat

Vygenerujeme si data, na kterých budeme prezentovat shlukování. Protože pracujeme se sítí bez učitele, postačí nám vstupní data a nepotřebujeme generovat žádná výstupní data.

In[5]:=
```
values = 10;
(* hodnoty ve {,} udávají x-ový a y-ový rozsah každého shluku *)
cluster1x = RandomReal[{-0.1, 0.1}, {values, 1}];
cluster1y = RandomReal[ {-0.1, 0.1}, {values, 1}];
cluster2x = RandomReal[{-0.1, 0.1}, {values, 1}];
cluster2y = RandomReal[ {0.9, 1.1}, {values, 1}];
cluster3x = RandomReal[{0.3, 0.5}, {values, 1}];
cluster3y = RandomReal[ {1.4, 1.6}, {values, 1}];
cluster4x = RandomReal[{0.9, 1.1}, {values, 1}];
cluster4y = RandomReal[ {2.1, 1.9}, {values, 1}];
cluster5x = RandomReal[{1.4, 1.6}, {values, 1}];
cluster5y = RandomReal[ {1.4, 1.6}, {values, 1}];
cluster6x = RandomReal[{1.9, 2.1}, {values, 1}];
cluster6y = RandomReal[ {1.1, 0.9}, {values, 1}];
cluster1 = Join[cluster1x, cluster1y, 2];
cluster2 = Join[cluster2x, cluster2y, 2];
cluster3 = Join[cluster3x, cluster3y, 2];
cluster4 = Join[cluster4x, cluster4y, 2];
cluster5 = Join[cluster5x, cluster5y, 2];
cluster6 = Join[cluster6x, cluster6y, 2];
inData = Join[cluster1, cluster2, cluster3, cluster4, cluster5, cluster6];
```

Vygenerovaná data jsou dvoudimenzionální a obsahují 6 jasně oddělených shluků.

Data si můžeme zobrazit zde.

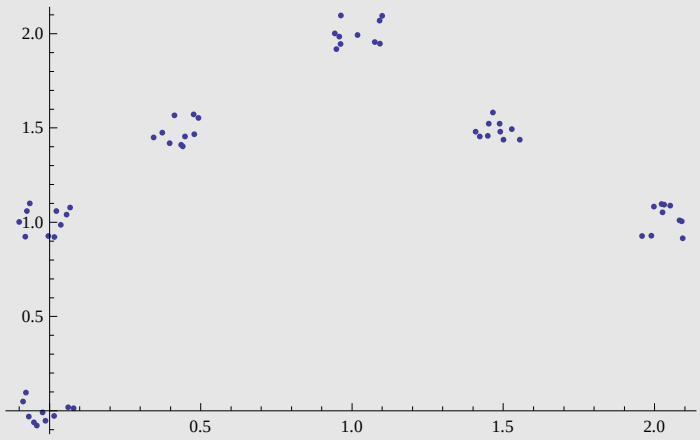
In[19]:=
```
inData
```

Out[19]=
```
{{0.0801371, 0.0138515}, {-0.0133842, -0.0522309}, {-0.0777816, 0.0967349},
 {-0.0228594, -0.00851835}, {0.0618867, 0.018003}, {-0.0687696, -0.0298561},
 {0.0155531, -0.0271162}, {-0.0873329, 0.0488711}, {-0.0512092, -0.0614278},
 {-0.0417654, -0.0780448}, {-0.0998226, 1.00242}, {0.0683901, 1.07761},
 {-0.065575, 1.09987}, {-0.0804564, 0.922928}, {0.0157775, 0.921475},
 {-0.00364633, 0.927098}, {0.037349, 0.985569}, {0.0229554, 1.06015},
 {0.0562783, 1.04058}, {-0.0745317, 1.06021}, {0.37239, 1.47466},
 {0.447559, 1.45457}, {0.344498, 1.4493}, {0.412644, 1.56756}, {0.397112, 1.41904},
 {0.440303, 1.40315}, {0.478278, 1.46665}, {0.435143, 1.41094},
 {0.476593, 1.57161}, {0.492363, 1.55422}, {1.01851, 1.99366}, {1.09094, 2.07013},
 {1.09255, 1.94794}, {1.07565, 1.95608}, {0.962042, 1.94613}, {0.947992, 1.91797},
 {0.963535, 2.09662}, {0.942717, 2.00161}, {0.95805, 1.98429}, {1.09969, 2.09527},
 {1.48767, 1.52336}, {1.52795, 1.494}, {1.4086, 1.48049}, {1.49012, 1.48093},
 {1.4518, 1.52369}, {1.4489, 1.45822}, {1.55411, 1.43814}, {1.42207, 1.45578},
 {1.50045, 1.43799}, {1.46591, 1.58304}, {2.02279, 1.09716}, {2.02658, 1.05259},
 {1.95864, 0.926335}, {1.98893, 0.928851}, {2.08248, 1.01062}, {2.05212, 1.08883},
 {2.09269, 0.915426}, {1.9979, 1.0824}, {2.08974, 1.0051}, {2.0315, 1.09418}}
```

Pro lepší představu o struktuře dat si je můžeme zobrazit v grafu.

In[20]:=
```
ListPlot[inData]
```

Out[20]=



Zpracování dat neuronovou sítí

Samoorganizující se neuronovou síť použijeme k hledání shluků ve vygenerovaných datech. Vytvoříme síť o šesti neuronech, kterou necháme na datech naučit. Po naučení sítě chceme aby ideálně každý neuron reprezentoval jeden shluk ve vstupních datech. Poté je možné spočítat hraniční přímky mezi jednotlivými neurony a tím de facto rozdělit do požadovaného počtu shluků celý vstupní prostor.

Náhodně inicializovaná síť

Nejprve vytvoříme neuronovou síť o šesti neuronech. Počet neuronů určuje kolik shluků se síť pokusí v datech najít.

In[21]:=
```
unsup = InitializeUnsupervisedNet[inData, 6];
```

General::obspkg :

BarCharts` is now obsolete. The legacy version being loaded may conflict with current Mathematica functionality.

See the Compatibility Guide for updating information. ≫

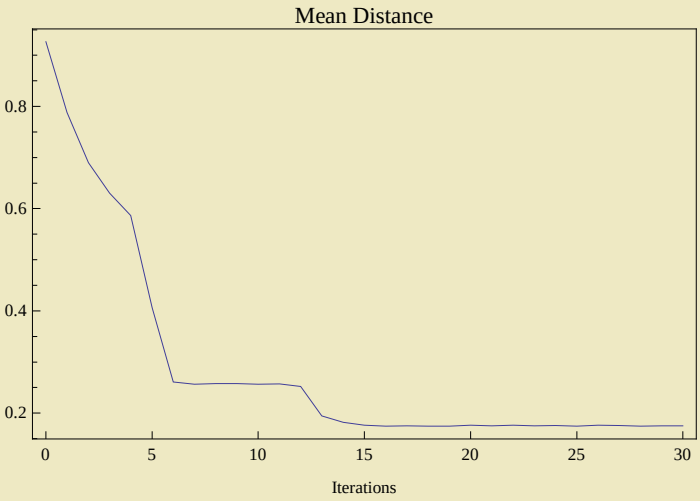
General::obspkg :

Histograms` is now obsolete. The legacy version being loaded may conflict with current Mathematica functionality.

See the Compatibility Guide for updating information. ≫

Tuto neuronovou síť naučíme na našich datech

In[22]:=
```
{unsup, fitrecord} = UnsupervisedNetFit[inData, unsup, 30, ReportFrequency → 1];
```



UnsupervisedNet::DeadNeuron :

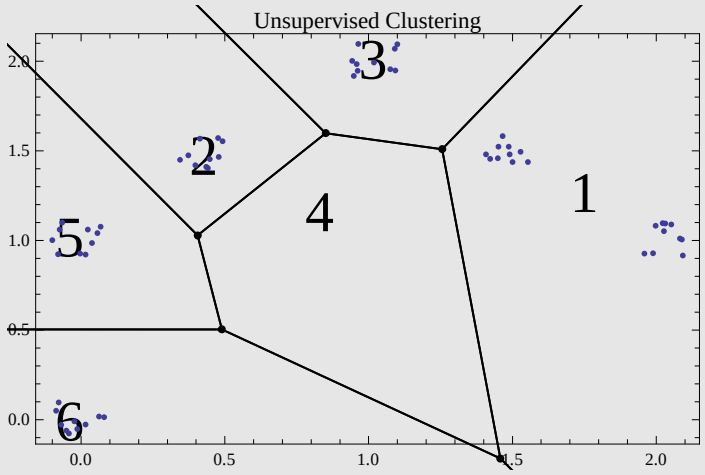
Some codebook vectors are not used by the data. They have 'died' during the training. Use UnUsedNeurons to identify them and decide if you want to remove them.

Vidíme že během učení sítě jeden z neuronů “zemřel”, tzn. není použit k rozlišení shluků v datech. Později se mu ještě budeme věnovat.

Ilustraci jaým způsobem byla data rozdělena do shluků nám poskytne následující graf. Pozice neuronu, který reprezentuje daný shluk je naznačena tučným číslem. Je také jasně vidět, který neuron je “mrtvý” a nepřináší nám žádný užitek.

In[23]:=
```
NetPlot[unsup, inData, CbvSymbol → {FontSize → 30}]
```

Out[23]=



Můžeme také jednoduše nechat “oklasifikovat” nový vektor (zadáme jeho x a y souřadnici).

In[24]:=
```
unsup[{0.2, 0.7}]
```

Out[24]=
```
{5}
```

Číslo, které nám síť vrátila znamená číselné označení shluku, kam vektor spadá (odpovídá číslům v grafu)

Nyní si vzpomeneme, že máme v síti nepotřebný neuron, budeme ho tedy chtít odstranit.
Nejdříve zjistíme o který neuron se jedná.

In[25]:=
```
UnUsedNeurons[unsup, inData]
```

Out[25]=
```
{4}
```

A poté ho odstraníme. Novou síť bez nepotřebného neuronu pojmenujeme unsup1.

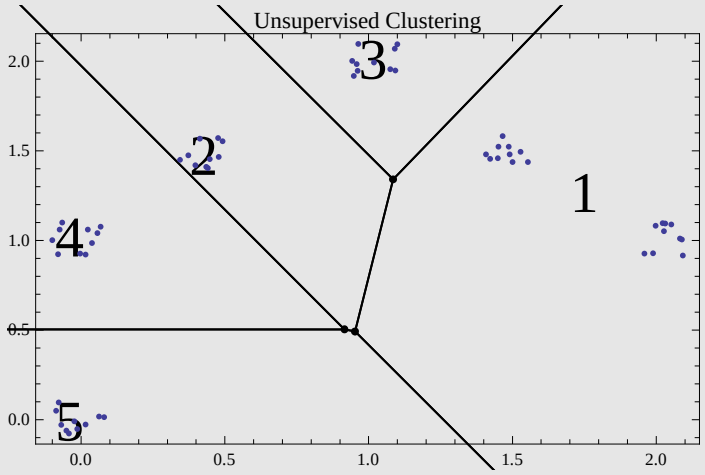
In[26]:=
```
unsup1 = NeuronDelete[unsup, %]
```

Out[26]=
```
UnsupervisedNet[{-Codebook Vectors-},
 {CreationDate → {2011, 5, 24, 15, 26, 26.9889388},
  AccumulatedIterations → 30, SOM → None}]
```

Podíváme se jakým způsobem odděluje síť jednotlivé shluky po odebrání nepotřebného neuronu.

In[27]:=
```
NetPlot[unsup1, inData, CbvSymbol → {FontSize → 30}]
```

Out[27]=



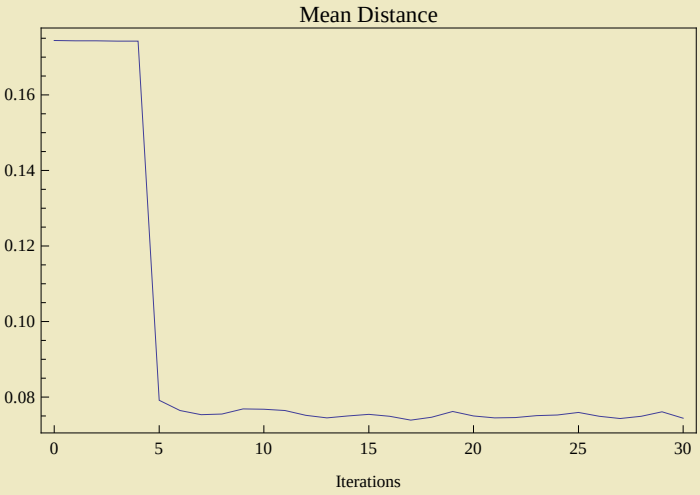
Síť inicializovaná metodou SOM

Protože vidíme, že rozdělení dat na shluky ještě není perfektní můžeme se pokust dosáhnout ještě lepšího rozdělení. Místo náhodné inicializace sítě můžeme použít SOM (samoorganizující mapu) k inicializaci sítě. Samoorganizující mapa se nechá přizpůsobit datům, na souřadnicích, kde skončí její neurony, jsou inicializovány neurony naší požadované sítě (naše síť není SOM).

In[28]:=
```
unsup2 = InitializeUnsupervisedNet[inData, 6, UseSOM → True,
    CriterionPlot → True, CriterionLog → True, Iterations → 30];
```

Nyní naučíme zinicializovanou síť na našich datech.

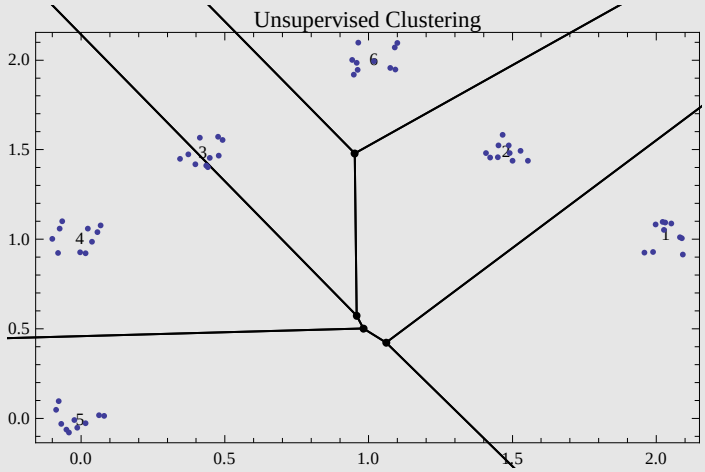
In[29]:=
```
{unsup2, fitrecord2} = UnsupervisedNetFit[inData, unsup2, 30];
```



A podíváme se jakým způsobem určuje shluky v datech.

In[30]:=
```
NetPlot[unsup2, inData]
```

Out[30]=



Tento přístup k inicializaci sítě nám může často pomoci dosáhnout lepších výsledků, než náhodná inicializace. Také často zabrání problému s “mrtvými” neurony.

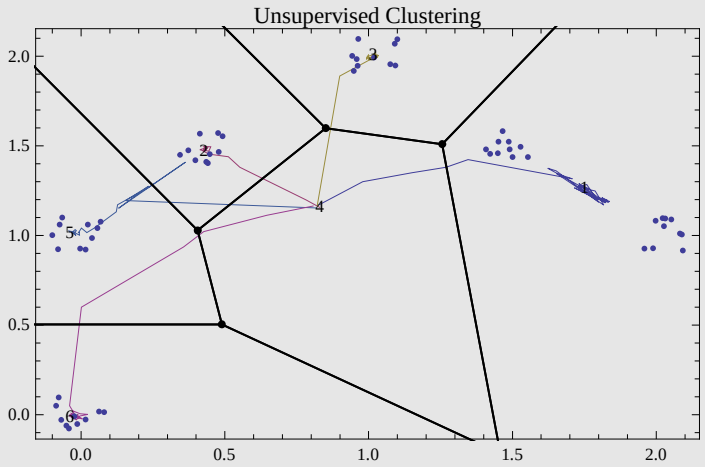
Průběh učení

- **Náhodně inicializovaná síť**

Ještě se můžeme podívat jak síť v průběhu učení dělila data do shluků a jak se měnila poloha jednotlivých neuronů. Barevné čáry ukazují pohyb jednotlivých neuronů v průběhu učení, čísla ukazují jejich konečnou polohu.

In[31]:=
```
NetPlot[fitrecord, inData]
```

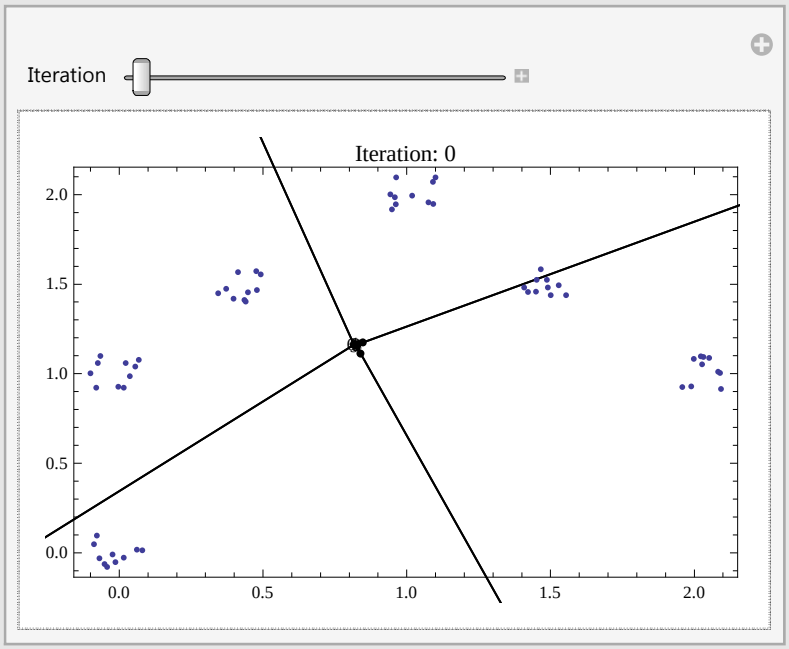
Out[31]=



Zde si můžete zobrazit průběh učení po jednotlivých iteracích. Pro správnou funkci je třeba mít vyhodnocené všechny výše se nacházející buňky.

In[32]:=
```
learningState = NetPlot[fitrecord, inData,
    DataFormat → DataMapArray, Intervals → 1][[1, 2, All, 1, 1]];
Manipulate[learningState[[a]], {{a, 1, "Iteration"}, 1, 31, 1}]
```

Out[33]=

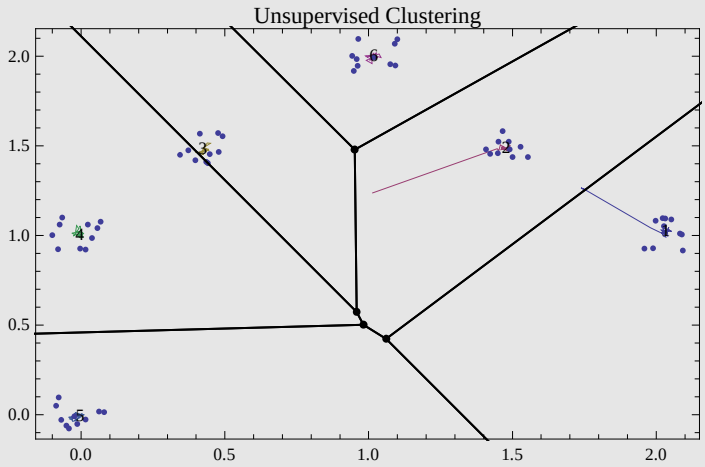


Síť inicializovaná metodou SOM

Z následujícího grafu je patrné, že síť zinicializovaná metodou SOM by poskytovala poměrně dobré výsledky bez nutnosti dalšího učení. Učení upravuje pouze pozici několika neuronů.

In[34]:=
```
NetPlot[fitrecord2, inData]
```

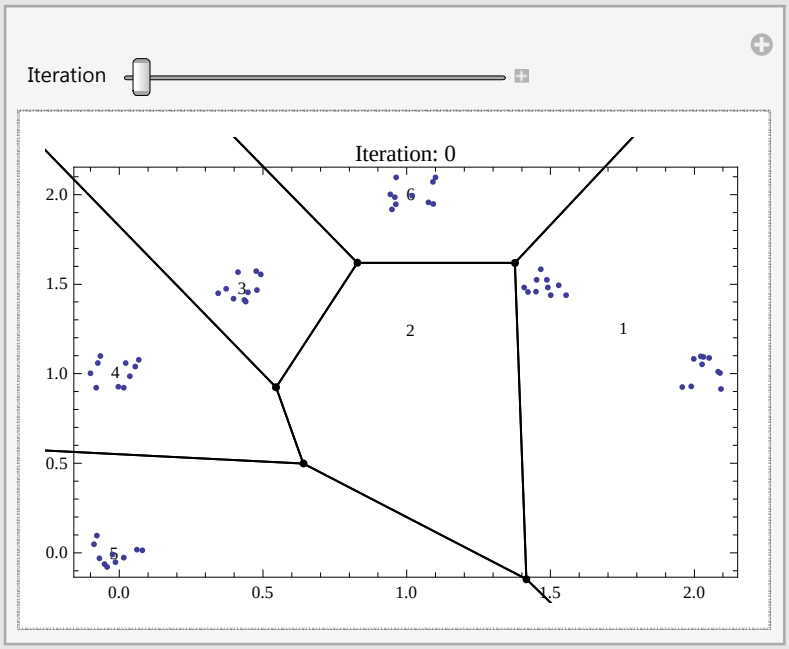
Out[34]=



Zde si můžete zobrazit průběh učení po jednotlivých iteracích. Pro správnou funkci je třeba mít vyhodnocené všechny výše se nacházející buňky.

In[35]:=
```
learningState2 = NetPlot[fitrecord2, inData,
    DataFormat → DataMapArray, Intervals → 1][[1, 2, All, 1, 1]];
Manipulate[learningState2[[a]], {{a, 1, "Iteration"}, 1, 31, 1}]
```

Out[36]=



Prohlášení

Tento text je součástí bakalářské práce Adama Činčury “Demonstrační aplikace pro podporu kurzu neuronových sítí” na FEL ČVUT 2011.