Kapitola 15 - LVQ a Iris data

Demonstrace použití LVQ neuronové sítě na datech z databáze UCI.

Načtení knihovny NeuralNetworks

Nejdříve načteme knihovnu neuronových sítí.

In[316]:=
```
<< 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[317]:=
```
Off[Remove::rmnsm]
```

Import dat

- **Načtení dat ze souboru**

Nastavíme si pracovní adresář na ten, kde máme uložen aktuální notebook, načítaná data musejí být ve stejném adresáři.

In[318]:=
```
SetDirectory[NotebookDirectory[]]
```

Out[318]=
```
D:\Dokumenty\BP
```

A načteme data.

In[319]:=
```
data = Import["iris.data"];
```

- **Načtení dat přímo z internetu**

Jiná možnost je importovat data přímo z internetu - příklad pro UCI databázi (stejná data jako v předchozím příkladu - načtení dat ze souboru).

In[320]:=
```
data =
  Import["http://ftp.ics.uci.edu/pub/machine-learning-databases/iris/iris.data"];
```

Předzpracování dat

Provedeme předzpracování dat pomocí stejného postupu, jaký je popsán v kapitole 5 - Dopředná síť a Iris data.

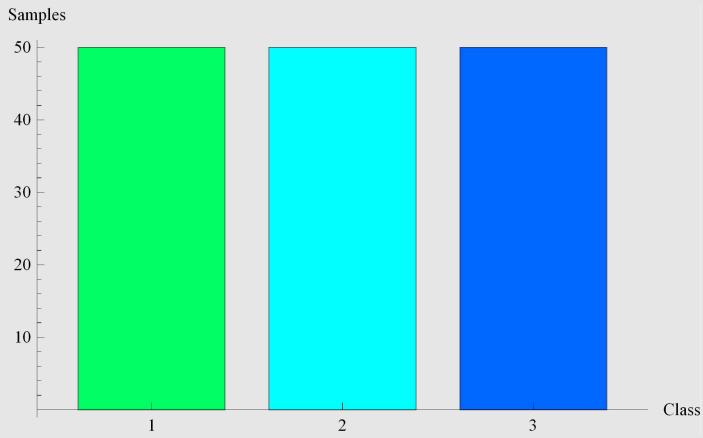
In[321]:=
```
data2 = Drop[data, -1];
inData = data2[[All, 1 ;; 4]];(*vstupní parametry*)
outDataTmp = data2[[All, 5]];(*výstupní parametr*)
outVal = Tally[outDataTmp][[All, 1]];
encode = MapIndexed[#1 -> Normal[SparseArray[#2 → 1, {Length[outVal]}]] &, outVal];
outData = Flatten[outDataTmp] /. encode;
```

Zpracování dat neuronovou sítí

Nejdříve se podíváme na rozložení jednotlivých skupin vzorků v datech. Následující příkaz vykreslí vstupní data a jejich množství v jednotlivých třídách.

In[327]:=
```
NetClassificationPlot[inData, outData,
 DataFormat -> BarChart, BarStyle -> {Hue[.4], Hue[.5], Hue[.6]}]
```

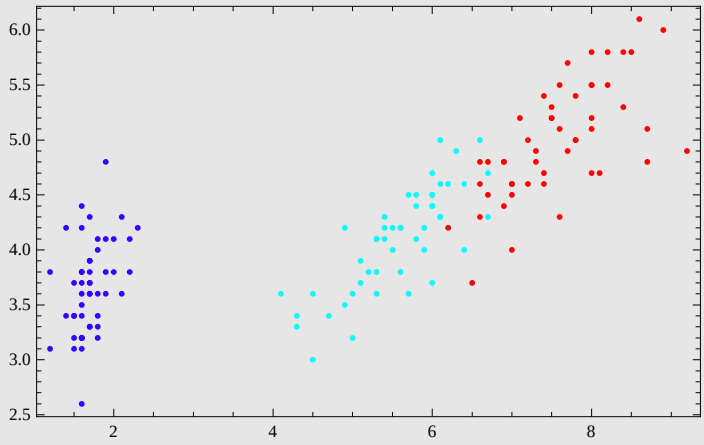
Out[327]=



Názornější pohled na 3 a vícedimenzionální data lze zobrazit pomocí roznásobení vstupních vektorů s vektory zobrazovaného prostoru. Na problém, jak nejlépe zvolit tyto vektory, je nejlepším řešením vyzkoušet nekolik různých variant a vybrat tu, ve které jsou data nejlépe separována.

In[328]:=
```
NetClassificationPlot[inData . {{0, 0}, {0, 1}, {1, 0}, {1, 1}},
 outData, PlotStyle → {Hue[.7], Hue[.5], Hue[.0]}]
```

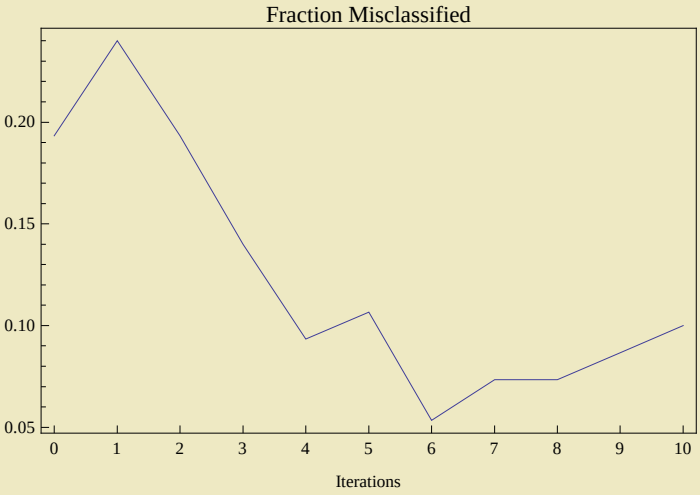
Out[328]=



Vstupní data namapujeme sítí, jejíž uspořádání můžeme zadat. Čím víc neuronů použijeme, tím bude lépe pokryt prostor vstupních dat. Číslo 5 udává počet neuronů. Poslední číslo v příkazu udává počet

iterací, není nutné ho uvádět, poté proběhne 6 iterací. Z vykresléného grafu je nejlépe poznat potřebná hodnota.

In[329]:=
```
{vq, fitrecord} = VQFit[inData, outData, 5, 10];
```



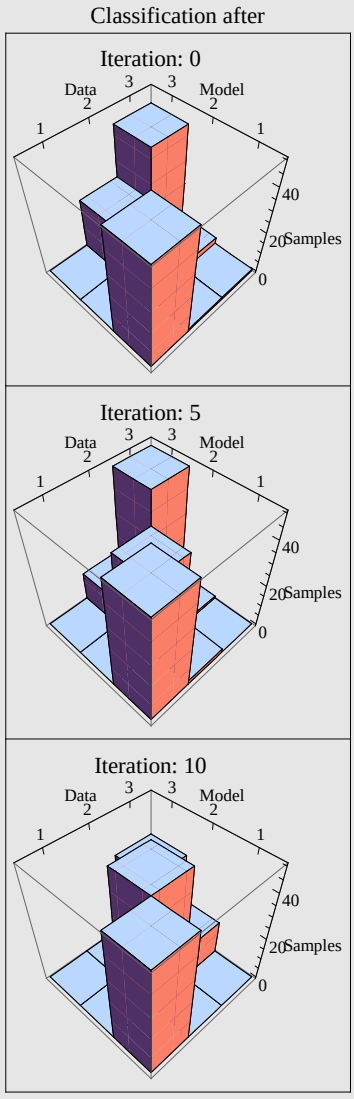
VQFit::DeadNeuron :

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.

Varovná hláška říká, že některé neurony nejsou využity pro klasifikaci. V další části se tedy podíváme, které to jsou a odstraníme je, abychom neměli zbytečně velkou síť. Předtím se však ještě podívejme na průběh učení. Síť, která dokonale odpovídá na vstupní data, má všechny vzorky na úhlopříčce.

In[330]:=
```
NetPlot[fitrecord, inData, outData, DataFormat → BarChart]
```

Out[330]=



Nyní odebereme nepotřebné neurony.
Nejprve si zobrazíme informace o síti. S těmito informacemi poté porovnáme síť s odstraněnými neurony.

In[331]:=
```
NetInformation[vq]
```

Out[331]=
```
VQ network for 3 classes with {2, 2, 1} codebook vectors per class, altogether
  5. It takes data vectors with 4 components. Created 2011-5-23 at 1:16.
```

Zjistíme, které neurony jsou nepoužité.

In[332]:=
```
UnUsedNeurons[vq, inData]
```

Out[332]=
```
{{2}, {}, {}}
```

Vybereme je.

In[333]:=
```
neuronyNaSmazani = Flatten[
  MapIndexed[
   {radek, #} & /@ #1 /. radek → #2[[1]] &,
   UnUsedNeurons[vq, inData]
  ], 1
 ]
```

Out[333]=
```
{{1, 2}}
```

A smažeme.

In[334]:=
```
vq = NeuronDelete[vq, neuronyNaSmazani]
```

Out[334]=
```
VQ[{-Codebook Vectors-},
 {CreationDate → {2011, 5, 23, 1, 16, 41.0967261}, AccumulatedIterations → 10}]
```

Nyní si znovu zobrazíme informace o síti a vidíme, že se odebrání povedlo.

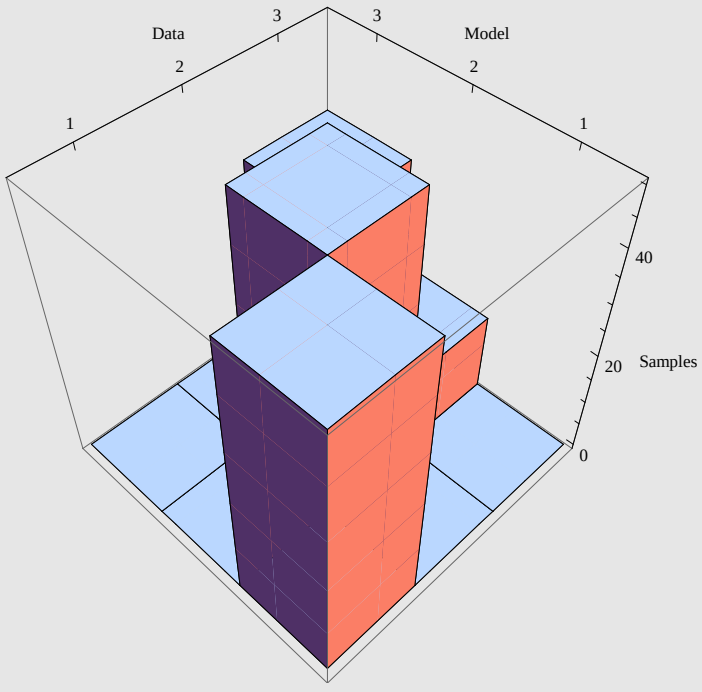
In[335]:=
```
NetInformation[vq]
```

Out[335]=
```
VQ network for 3 classes with {1, 2, 1} codebook vectors per class, altogether
  4. It takes data vectors with 4 components. Created 2011-5-23 at 1:16.
```

A takto vypadá klasifikace naučené sítě.

In[336]:=
```
NetPlot[vq, inData, outData, CbvSymbol → {FontSize → 30}]
```

Out[336]=



Získaná síť může být použita na mapování jakýchkoliv vektorů (se správnou dimenzí, shodnou se vstupními vektory sítě). Výstup ukazuje, ke kterému referenčnímu vektoru má nejblíže. V příkladu jsou použity vektory za vstupních dat, obecně se dají použít jakákoliv data.

In[337]:=
```
vq[{{6.4, 2.9, 4.3, 1.3}, {6.9, 3.2, 5.7, 2.3}}]
```

Out[337]=
```
{{0, 1, 0}, {0, 0, 1}}
```

Další příkazem zjistíme procentuální neúspěšnost klasifikace. Příkaz vrátí procento chybně klasifikovaných vektorů.

In[338]:=
```
VQPerformance[vq, inData, outData] * 100
```

Out[338]=
```
10.
```

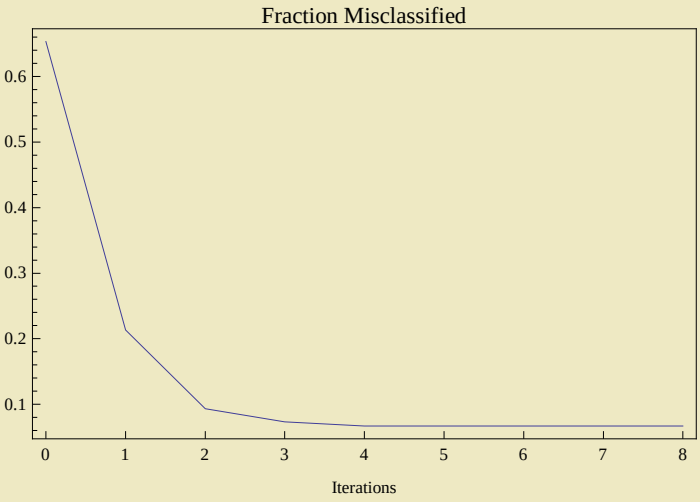
Dále si můžeme nechat zobrazit průměrnou vzdálenost mezi vstupními vektory a váhovými vektory. Čím menší je hodnota, tím lépe odpovídá síť datům.

In[339]:=
```
VQDistance[vq, inData]
```

Out[339]=
```
0.67307
```

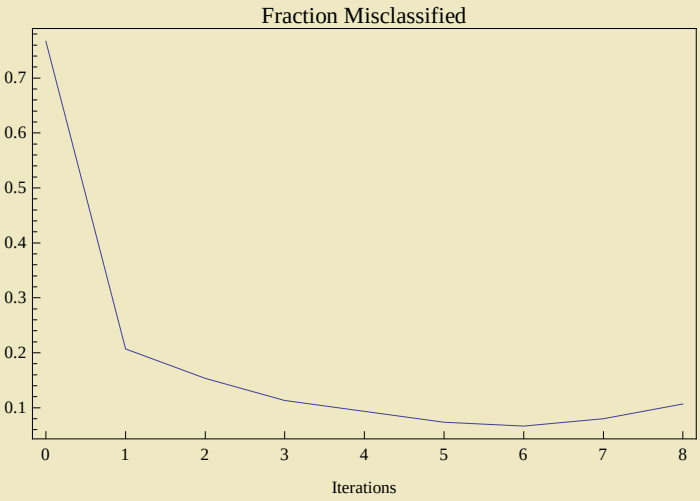
Pokud je pokrytí už velmi dobré, můžeme také nechat projít algoritmus učení opakovaně. Při volbě Recursive -> False je délka kroku učícího algoritmu nastavena na jedničku pro všechny iterace, což se hodí právě pro situace, kdy jsou váhové vektory blízko svým optimálním hodnotám. Pro vektory vzdálené svému optimu může být tento způsob nestabilní. Zatímco při Recursive -> True (výchozí hodnota) je délka kroku v prvních iteracích malá, takže váhové vektory najdou správnou orientaci. Poté je krok zvětšen, pro urychlení pokrytí. Od této vysoké hodnoty se délka kroku opět pomalu snižuje. Pokrytí je zaručeno, když se délka kroku blíží nule.
Nejdříve se podíváme jak dopadne nenaučená síť s jednotlivými volbami.

In[340]:=
```
{f, fiterecord} = VQFit[inData, outData, 4, 8, Recursive → False];
{t, fiterecord} = VQFit[inData, outData, 4, 8, Recursive → True];
```



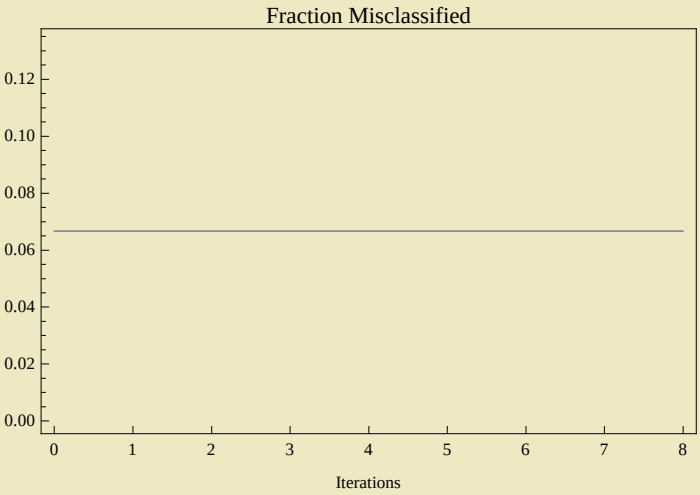
VQFit::DeadNeuron :

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.



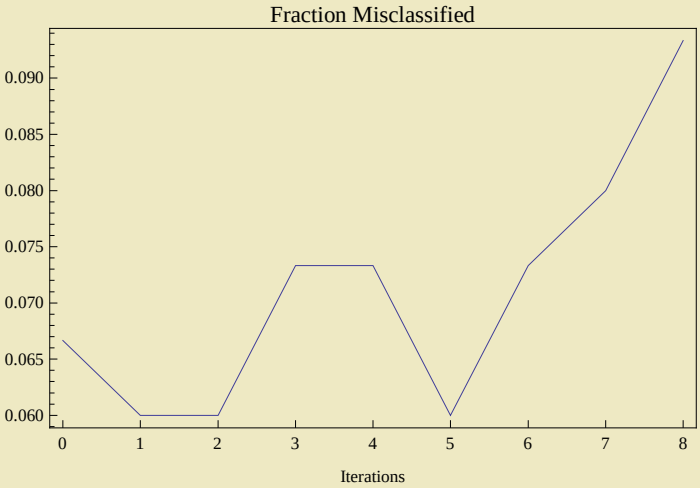
Doučení sítě, která byla vytvořena s parametrem Recursive -> False, nejprve opět s Recursive -> False, poté s hodnotou True.

In[342]:=
```
ff = VQFit[inData, outData, f, 8, Recursive → False];
ft = VQFit[inData, outData, f, 8, Recursive → True];
```



VQFit::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.

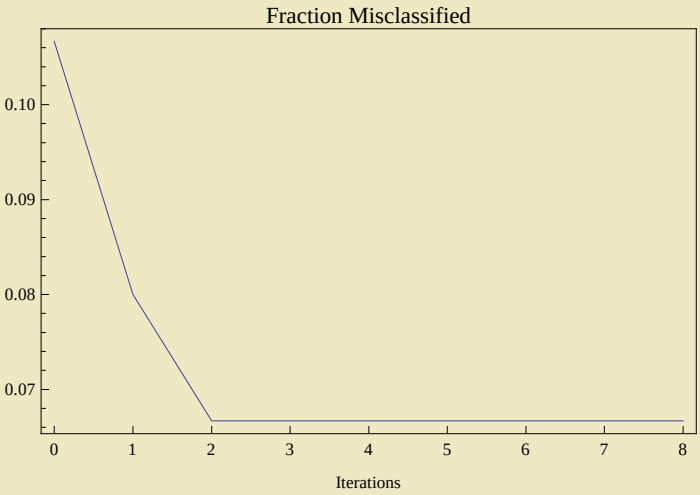


VQFit::DeadNeuron :

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.

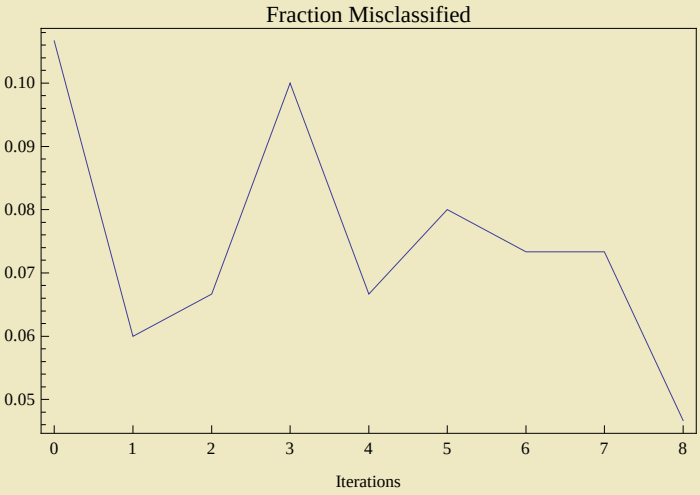
A doučení sítě, která byla vytvořena s parametrem Recursive -> True, nejprve Recursive -> False, poté s hodnotou True.

In[344]:=
```
tf = VQFit[inData, outData, t, 8, Recursive → False];
tt = VQFit[inData, outData, t, 8, Recursive → True];
```



VQFit::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.



VQFit::DeadNeuron :

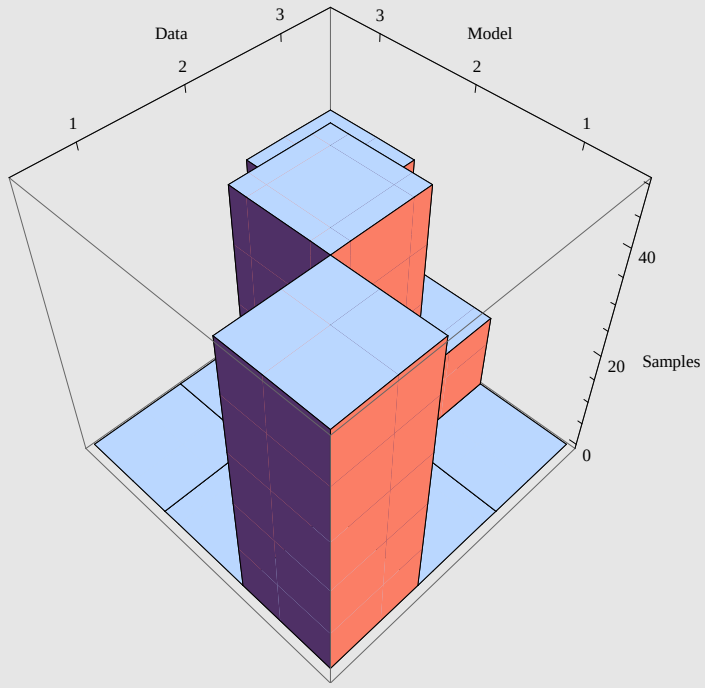
Some codebook vectors are not used by the data. They have 'died' during the training. Use UnUsedNeurons to identify them and decide if you want to remove them.

Vizualizace výstupu sítě

BarChart ve 3D zobrazení ukazuje správně klasifikované vektory a jejich počet na diagonále. Nesprávně klasifikované vektory jsou mimo ni.

In[346]:=
```
NetPlot[vq, inData, outData, DataFormat → BarChart]
```

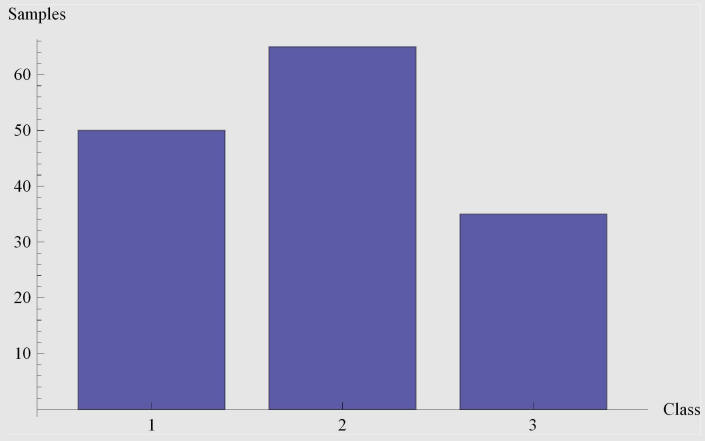
Out[346]=



2D BarChart nejlépe vypovídá o tom do které třídy naše síť klasifikuje kolik dat.

In[347]:=
```
NetPlot[vq, inData, DataFormat → BarChart]
```

Out[347]=



Zobrazení pomocí tabulky také vypovídá o rozložení vektorů do jednotlivých tříd. Vidímě díky ní také jaké data síť do které třídy klasifikuje. 2:50 a 3:12 znamená že je do třídy 2 klasifikováno 50 vektorů třídy 2 (správně oklasifikováno) a 12 vektorů třídy 3 (chybně oklasifikovány).

In[348]:=
```
NetPlot[vq, inData, outData, DataFormat -> Table]
```

Out[348]=

VQ Classification

	3:15	
1:50	2:50	3:35

Prohlášení

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