Рачунарска интелигенција

Вештачке неуронске мреже

Александар Картељ

kartelj@matf.bg.ac.rs

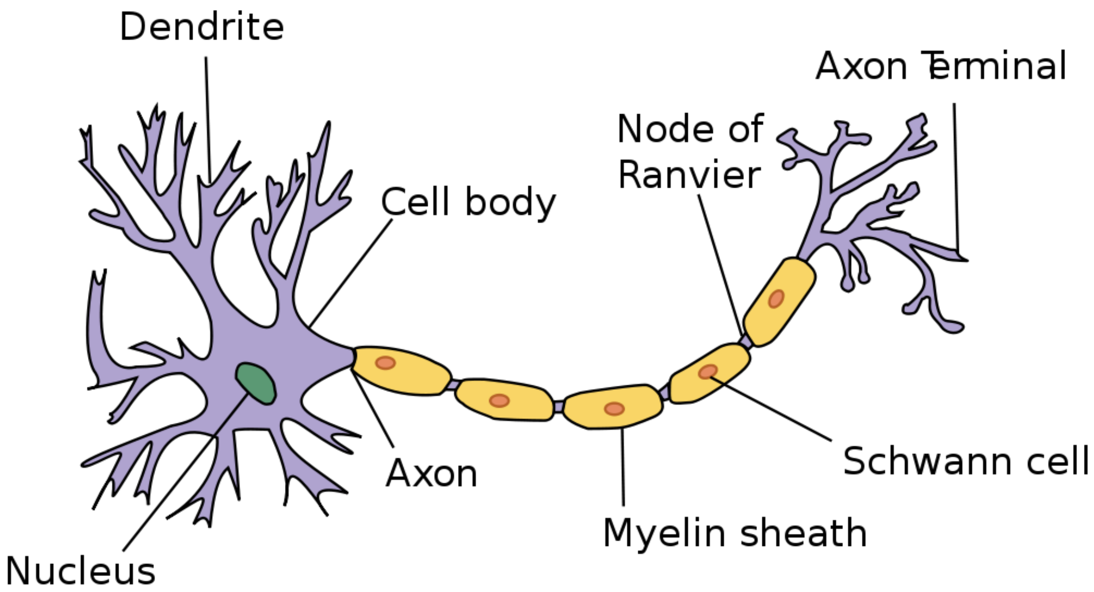
Датум последње измене: 16.10.2019.

Вештачке неуронске мреже

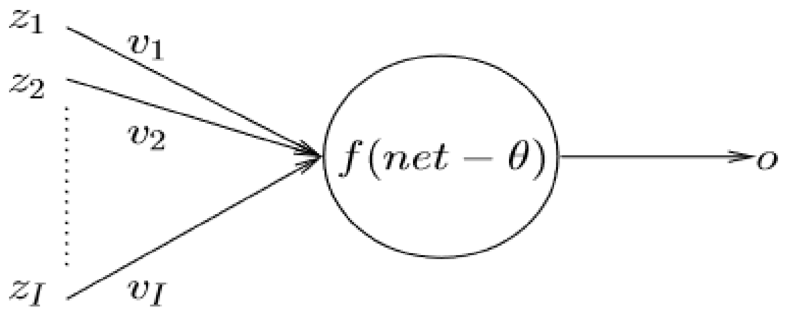
Вештачки неурон

Биолошки неурон

- ~10^{11} неурона
- Синапса повезује два неурона
    - Заслужна за памћење
    - Сваки импулс у синапси изазива лучење мале количине неуротрансмитера
    - Синапса може да поспеши или инхибира импулс
    - Она тип реакције памти током времена
- Неурони се не регенеришу као остале ћелије
    - То у комбинацији са синапсама омогућава памћење



Вештачки неурон



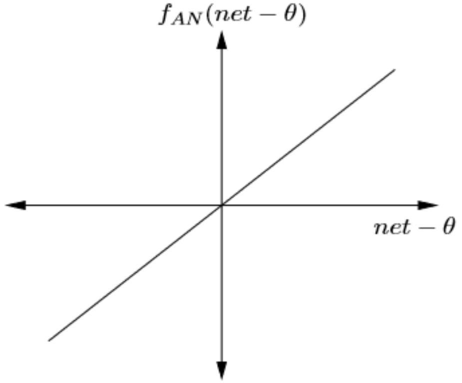
$$f_{AN} : \mathbb{R}^I \rightarrow [0, 1]$$

$$f_{AN} : \mathbb{R}^I \rightarrow [-1, 1]$$

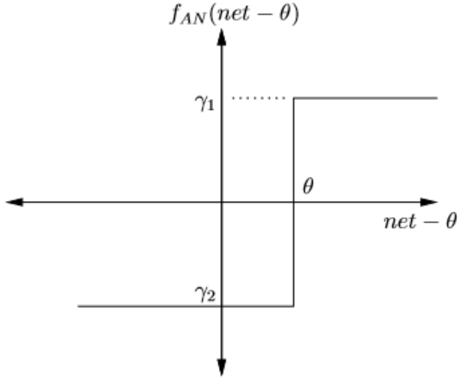
$$net = \sum_{i=1}^{I} z_i v_i \qquad net = \prod_{i=1}^{I} z_i^{v_i}$$

- McCulloch и Pitts (енг. Threshold Logic Unit – TLU)
- Омогућава апроксимацију нелинеарне функције f_{AN} слика улазне сигнале у излазни
- I – број улазних сигнала
- **z –** улазни сигнали
- **v –** тежине придружене улазним сигналима (симулација синапсе)
    - Позитивна тежина = ексцитација
    - Негативна тежина = инхибиција
- o – излазни сигнал
- net базиран на производу омогућава већи информациони капацитет

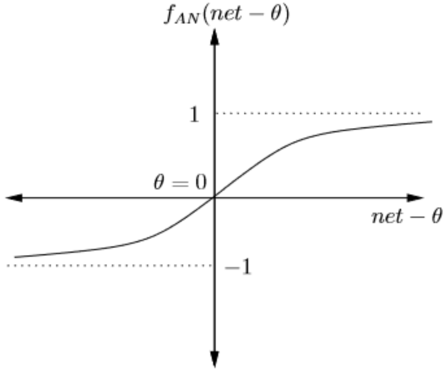
Функција активације



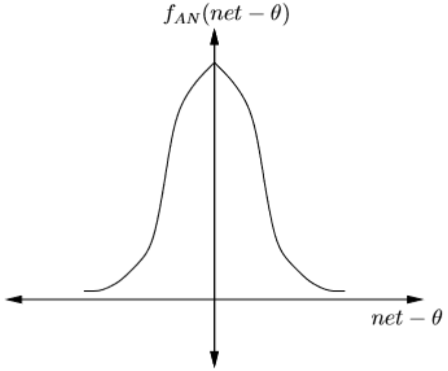
(a) Linear function



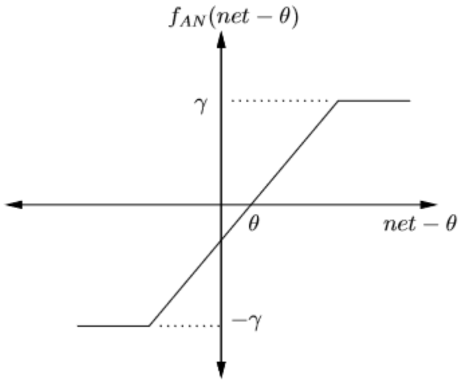
(b) Step function



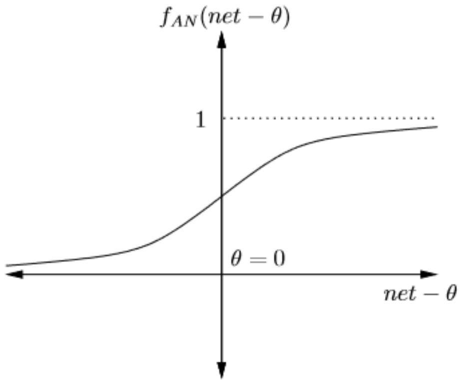
(e) Hyperbolic tangent function



(f) Gaussian function

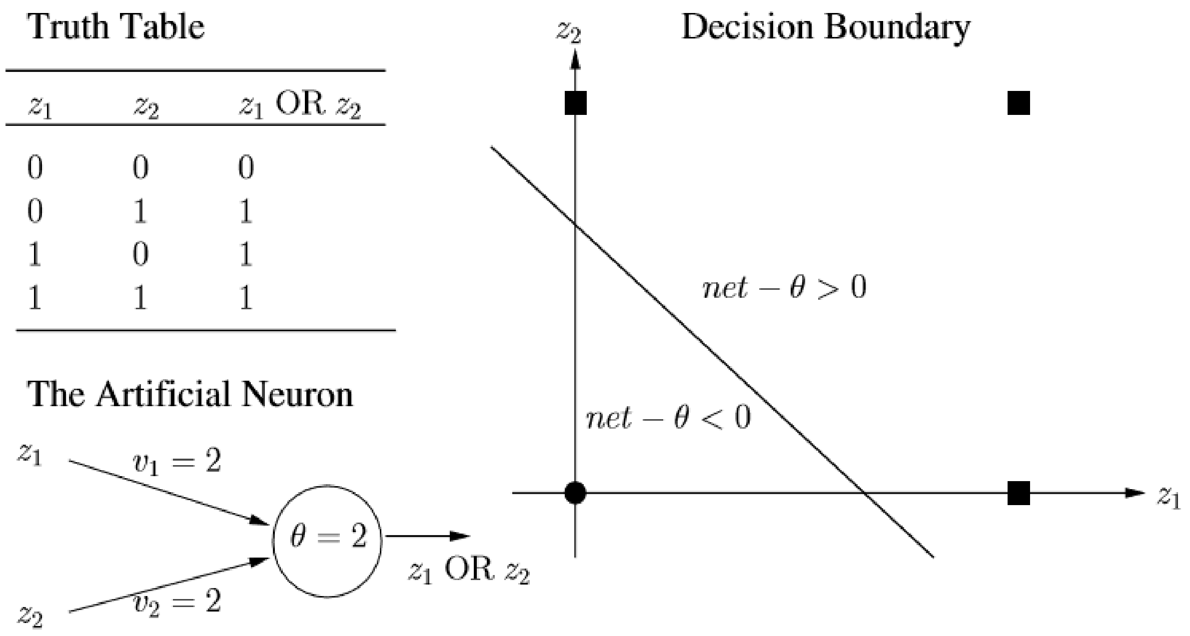


(c) Ramp function



(d) Sigmoid function

Линеарна раздвојивост

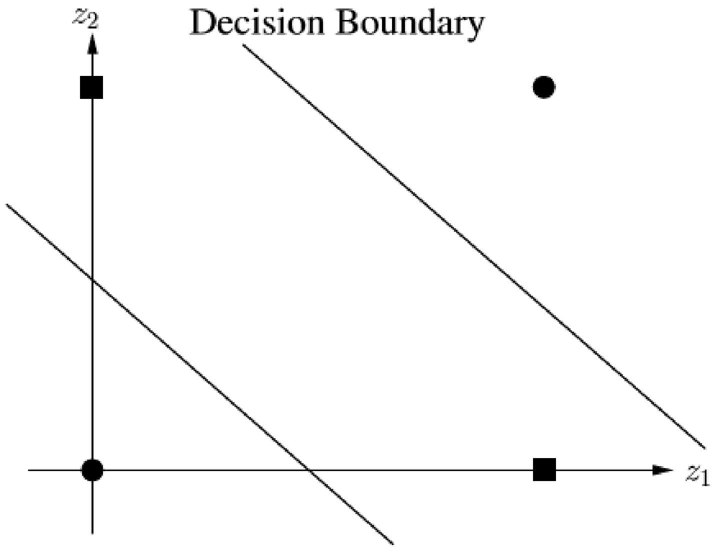


- Омогућава линеарну раздвојивост без грешке
- Постављање хиперравни која раздваја улазне податке на оне са излазом испод и изнад неког прага
- Слика приказује хиперраван која одговара функцији логичке дисјункције

Нелинеарна раздвојивост

Truth Table

z_1	z_2	z_1 XOR z_2
0	0	0
0	1	1
1	0	1
1	1	0



- Реализација ексклузивне дисјункције захтева постојање средишњег слоја са два неурона
- Ово је пример проблема који није линеарно раздвојив

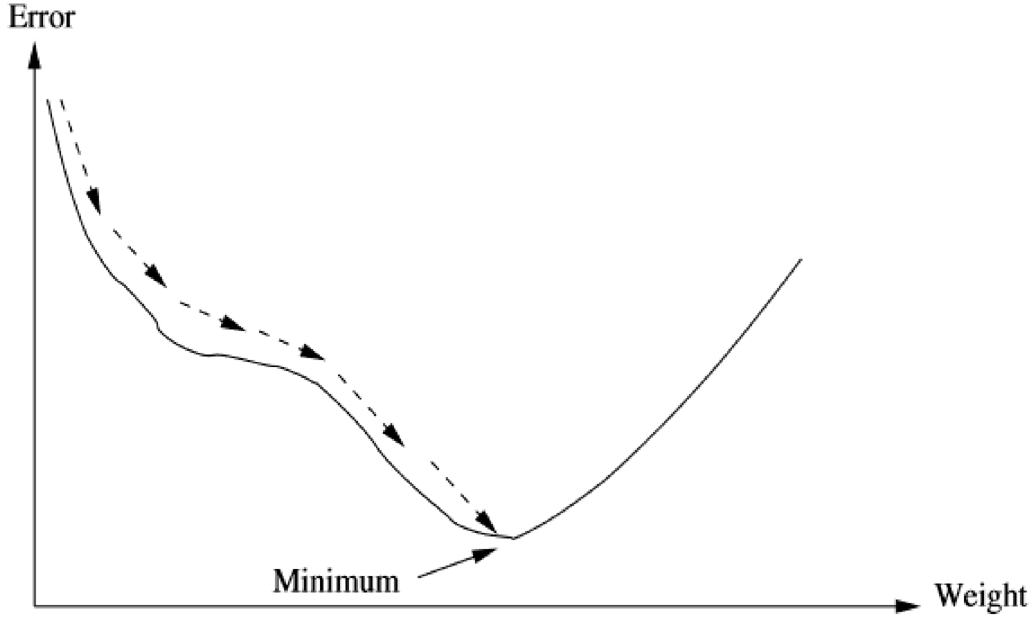
Учење градијентним спустом

- Вештачки неурон апроксимира функцију описану улазно излазним сигналима подешавањем тежина **v** и параметра θ
- Скаларни параметар θ се може придружити вектору тежина **v** ради елегантније нотације
- Апроксимација се своди на минимизацију укупне грешке:

$$\mathcal{E} = \sum_{p=1}^{P_T} (t_p - o_p)^2$$

- Где t_p и o_p представљају редом циљну и апроксимирану вредност излазног сигнала, а Pr број података спроведених на улаз.

Учење градијентним спустом (2)



- Правило градијентног спуста омогућава итеративно ажурирање тежина и дефинисано је као:

$$v_i(t) = v_i(t-1) + \Delta v_i(t)$$

$$\Delta v_i(t) = \eta(-\frac{\partial \mathcal{E}}{\partial v_i}) \qquad \frac{\partial \mathcal{E}}{\partial v_i} = -2(t_p - o_p)z_{i,p}$$

$$v_i(t) = v_i(t-1) + 2\eta(t_p - o_p)z_{i,p}$$

- Где је η параметар брзине учења.

Пример учења градијентним спустом

- Градијентним спустом научити тежине мреже тако да правилно класификује тачке A(2,1) и C(0.5,0.5) као једну класу,
  а тачку B(-1,-1) као другу класу.
- Претпоставити да су иницијалне тежине w1=2 и w2=3 док је θ=3.
- Брзина учења је 1.

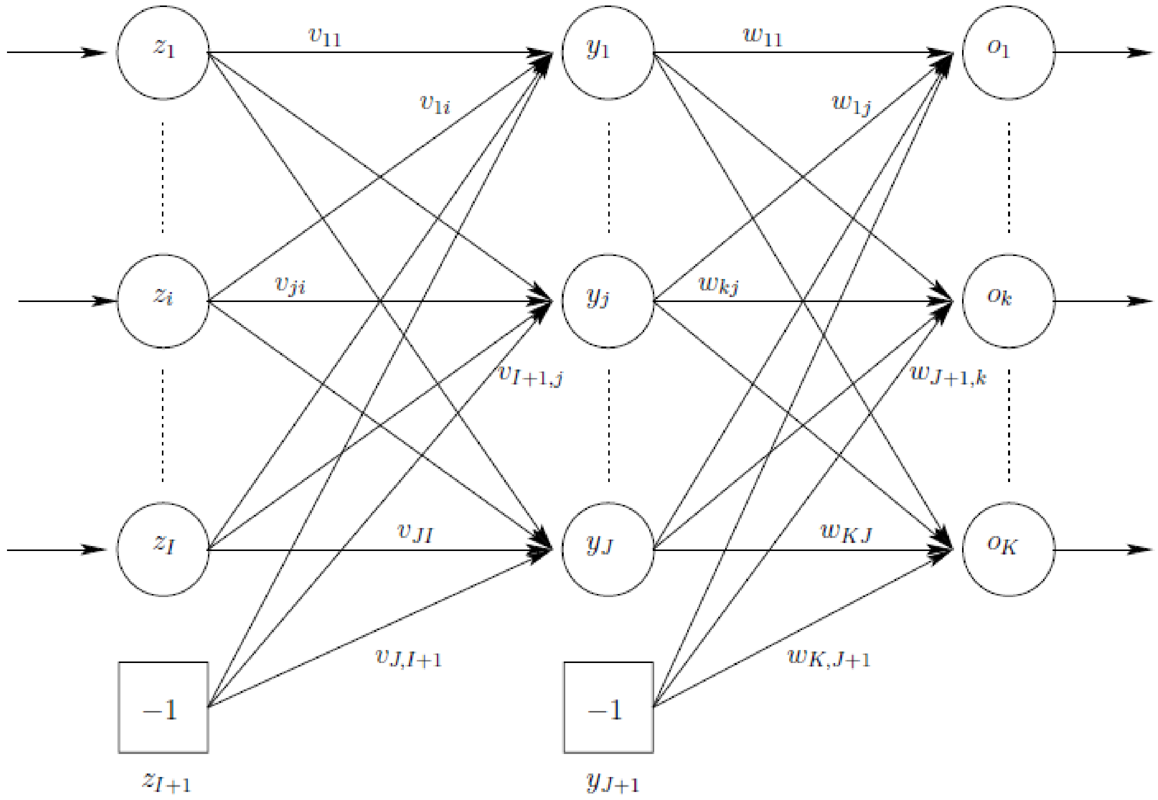
Вештачке неуронске мреже

Надгледано учење вештачке неуронске мреже

Учење неуронских мрежа

- Појединачни вештачки неурон омогућава учење само линеарно раздвојивих функција
  - Груписање неурона у мреже то омогућава
  - Учење оваквих мрежа је и значајно комплексније и рачунарски захтевно
  - Надгледано и ненадгледано учење
- Надгледано учење захтева скуп података за тренинг
  - Сваки податак (вектор променљивих) има своју придружени циљну променљиву

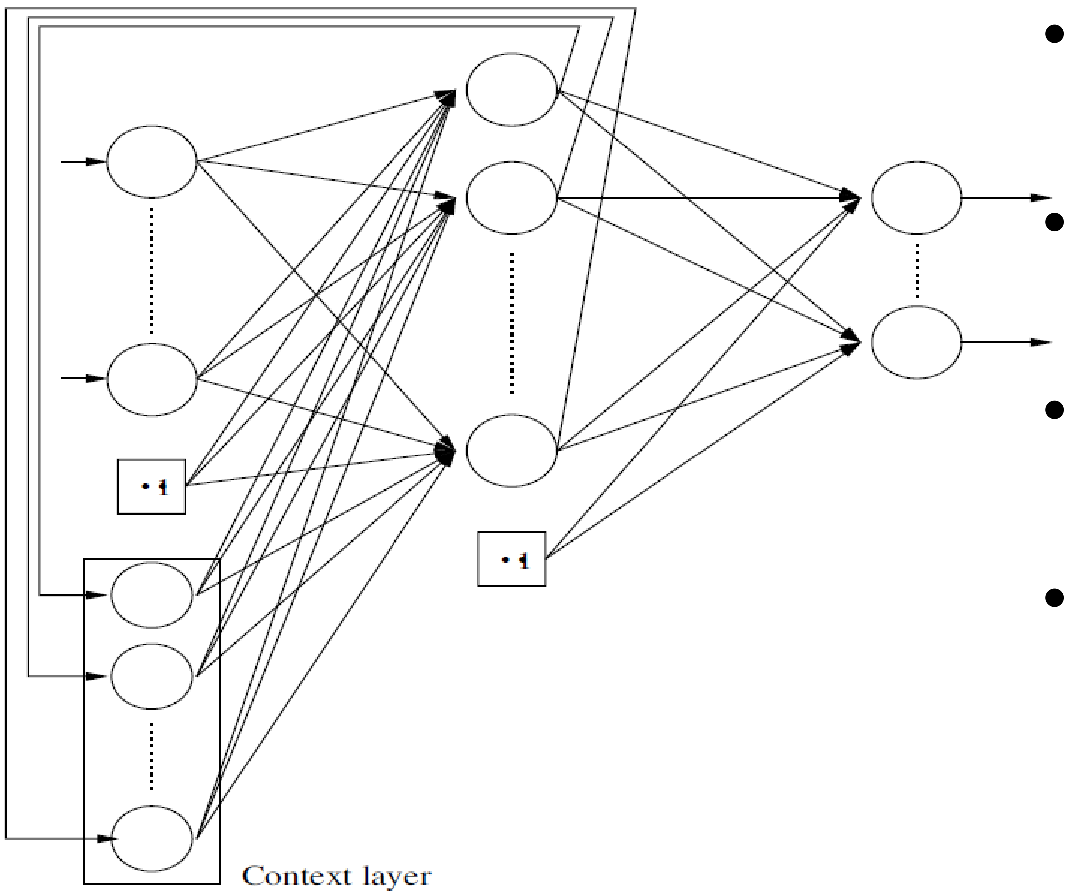
Мреже са пропагацијом унапред (FFNN)



- енг. Feedforward neural network
- Најмање три слоја:
  улазни, средњи и излазни
- Излаз се рачуна помоћу једног проласка кроз мрежу

$$
\begin{aligned}
o_{k,p} &= f_{o_k}(net_{o_{k,p}}) \\
&= f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} f_{y_j}(net_{y_{j,p}})\right) \\
&= f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} f_{y_j}\left(\sum_{i=1}^{I+1} v_{ji} z_{i,p}\right)\right)
\end{aligned}
$$

Рекурентне неуронске мреже

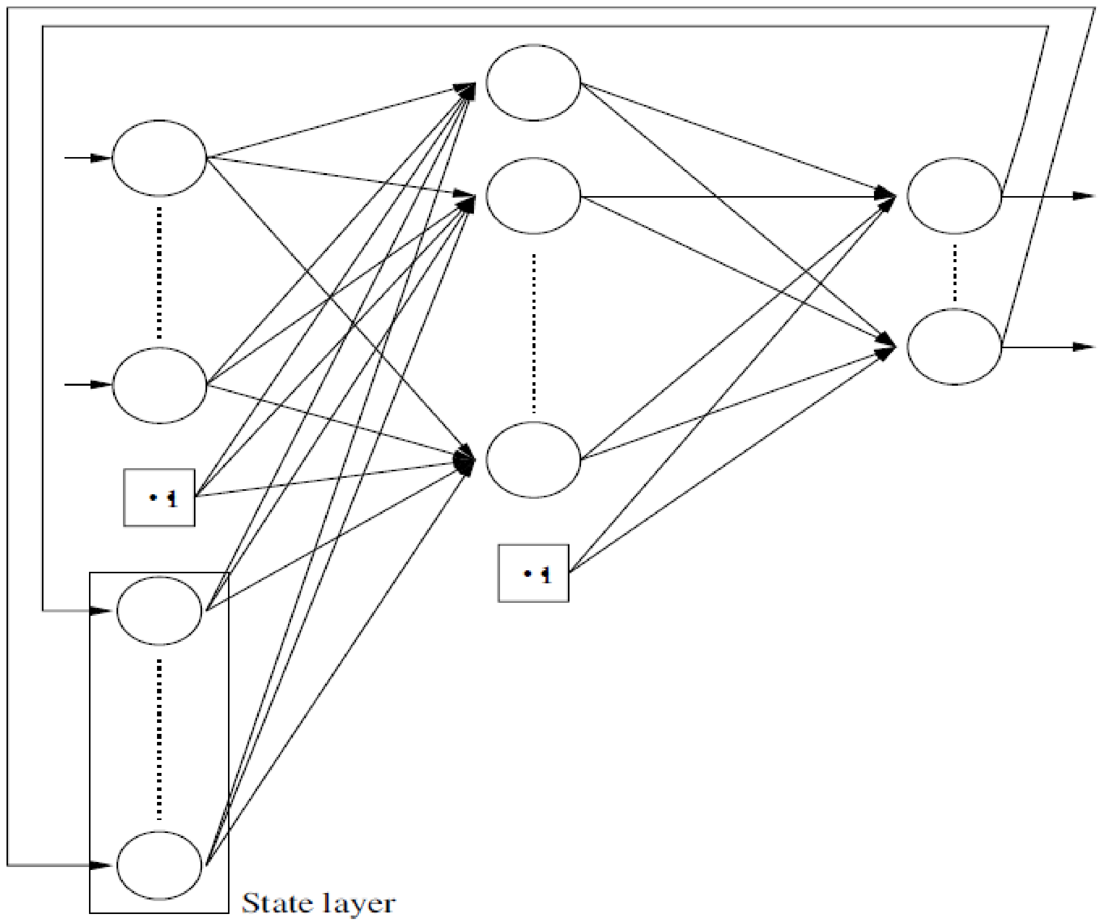


- Elman SRNN
  (енг. Simple recurrent neural network)
- Копија скривеног слоја се враћа на улаз (контекстни слој)
- Циљ је употреба претходног стања мреже
- Омогућава нпр. учење темпоралних зависности

$$o_{k,p} = f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} f_{y_j}\left(\sum_{i=1}^{I+1+J} v_{ji} z_{i,p}\right)\right)$$

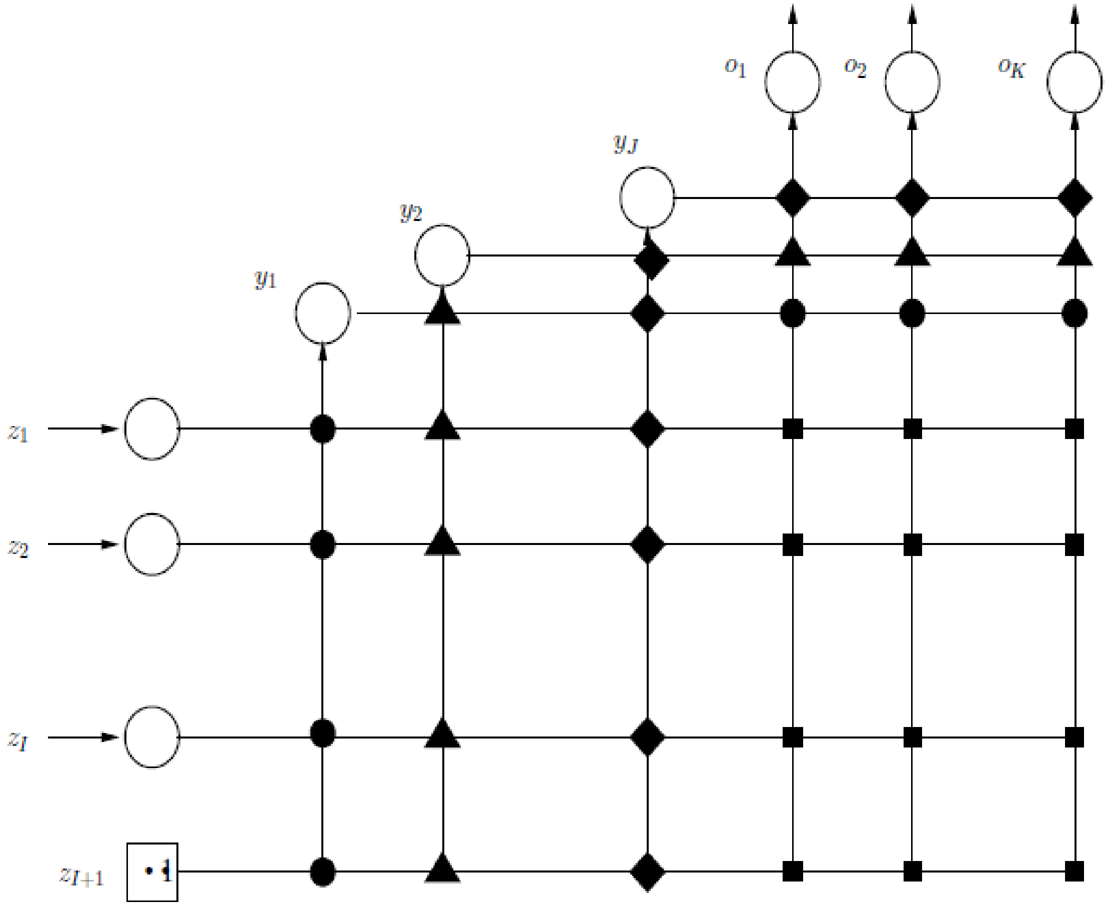
$$(z_{I+2,p}, \cdots, z_{I+1+J,p}) = (y_{1,p}(t-1), \cdots, y_{J,p}(t-1)).$$

Рекурентне неуронске мреже (2)



- Jordan SRNN
- Копија излазног слоја се спроводи на улаз (тзв. слој стања)

Каскадне неуронске мреже



- CNN (енг. Cascade NN)
- Сви улази спојени са свим скривеним и свим излазним елементима
- Елементи средњег слоја спојени са свим излазима и свим наредним елементима средњег слоја

Правила надгледаног учења

- Нека је дат коначан скуп уређених парова улазних вредности и придружених циљних вредности:

  $$D = \{dp = (\mathbf{z}_p, \mathbf{t}_p) | p = 1, \cdots, P\}$$

- Где су $z_{i,p}$, $t_{k,p} \in \mathrm{R}$ за $i = 1, \cdots, I$ и $k = 1, \cdots, K$
- I је број улазних сигнала
- K је број излазних сигнала
- P је број тренинг података
- Тада се може представити следећа зависност:

  $$\mathbf{t}_p = \mu(\mathbf{z}_p) + \zeta p$$

- Где је $\mu(\mathbf{*})$ непозната циљна функција, а ζp шум

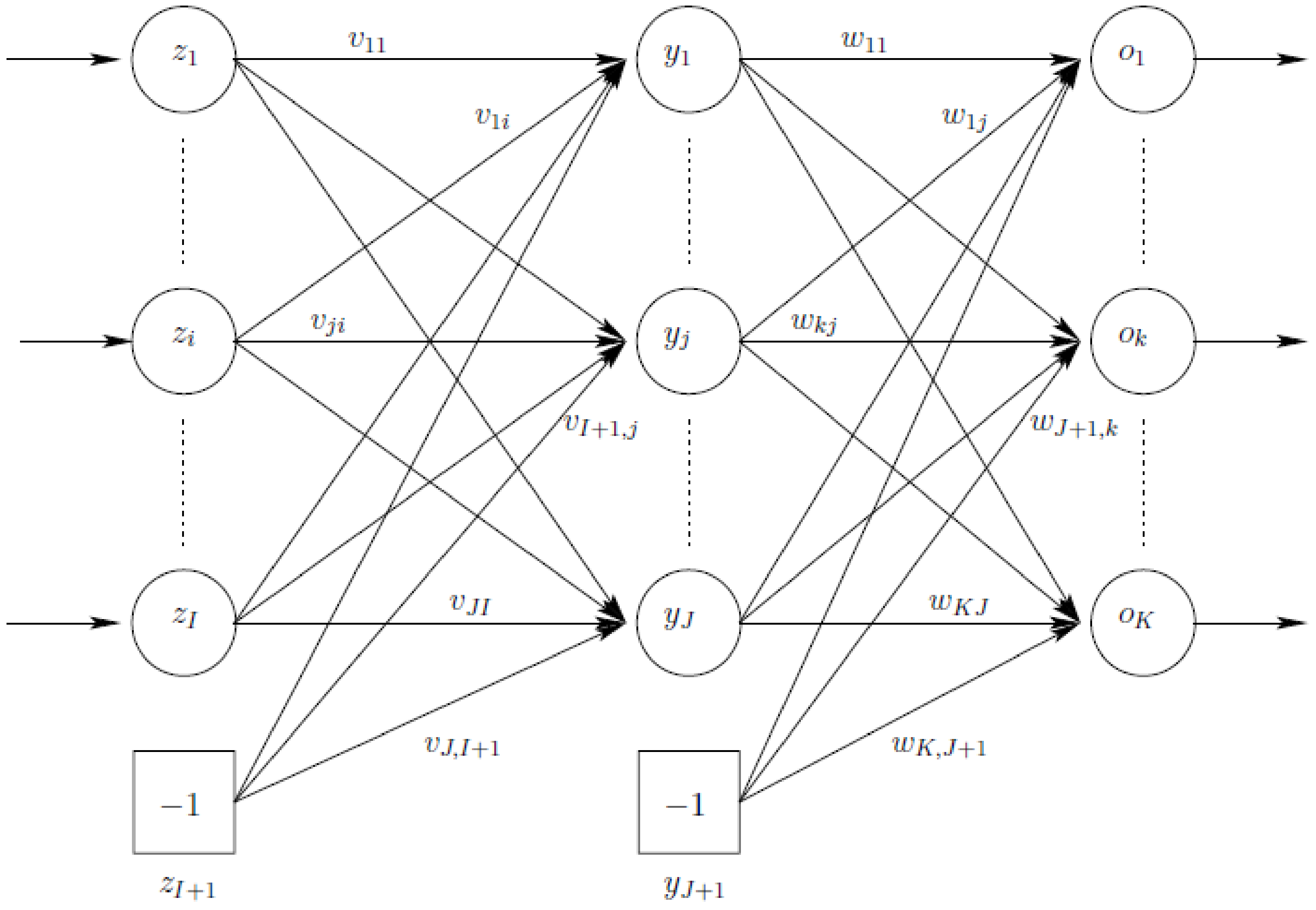
Правила надгледаног учења (2)

- Циљ учења је апроксимирати дату функцију $\mu(*)$ на основу података из *D*
- Полазни скуп *D* се обично дели на три дисјунктна подскупа:
  - D_T – тренинг скуп за апроксимацију
  - D_V – скуп за валидацију (меморизација)
  - D_G – скуп за тестирање (процена квалитета уопштавања)
- Током фазе учења минимизује се емпиријска грешка подешавањем *W*:

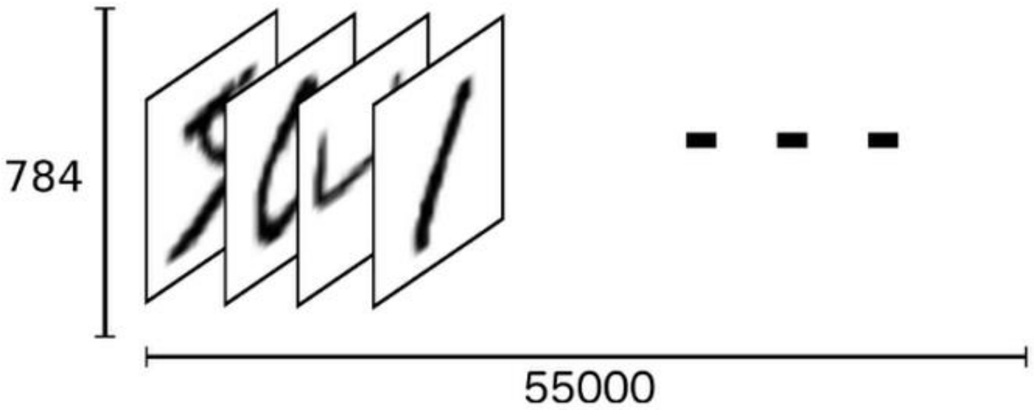
$$\mathcal{E}_T(D_T; \mathbf{W}) = \frac{1}{P_T} \sum_{p=1}^{P_T} (F_{NN}(\mathbf{z}_p, \mathbf{W}) - \mathrm{t}_p)^2$$

- Постоје разне технике за оптимизацију овог типа:
  - Методе локалне оптимизације: градијентни спуст нпр.
  - Методе глобалне оптимизације: метахеуристике нпр.
- Изазови: преприлагођавање и потприлагођавање?

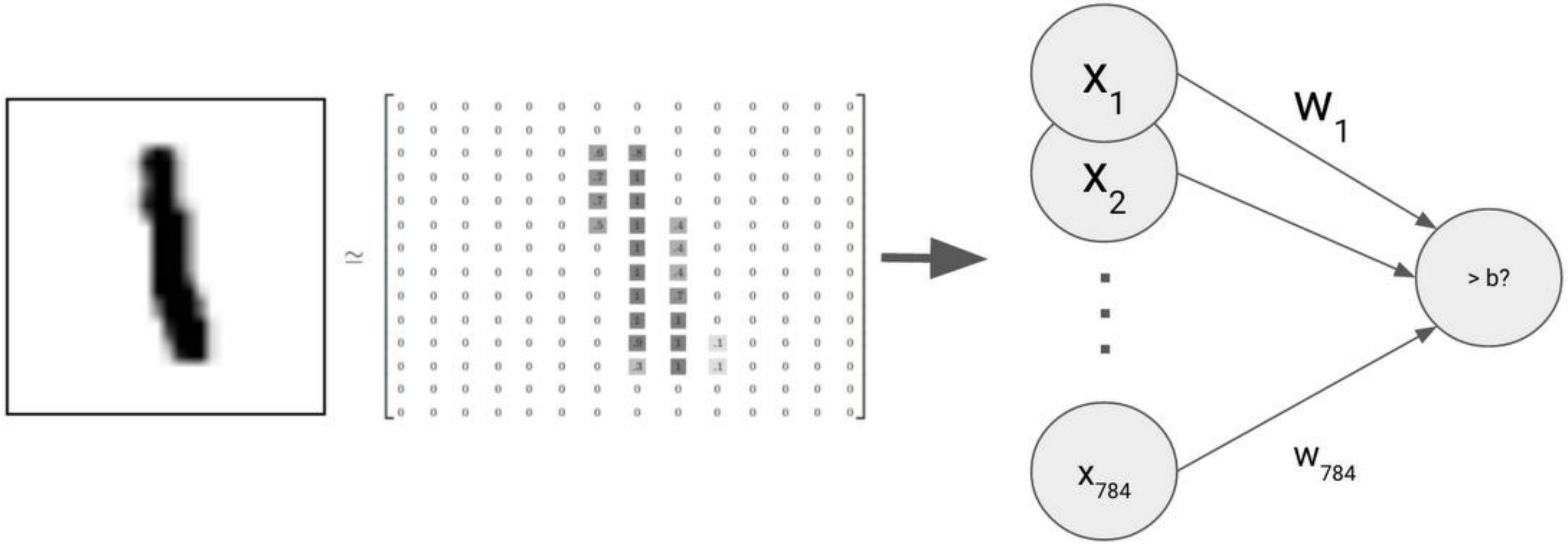
Подсећање на структуру мреже



Пример – препознавање цифара



- Улаз је матрица пиксела 28x28
- На излазу је 10 сигнала – сваки одговара једној цифри



Градијентни спуст за учење NN

- Састоји из две фазе:
  1. Пропагација сигнала унапред, једноставно рачунање сигнала за FFNN
  2. Пропагација грешке уназад: сигнал грешке се шаље назад ка улазном слоју при чему се врши измена тежинских коефицијената
- Нека је сума квадратна грешка (енг. Sum squared error - SSE) узета за функцију циља минимизације:

$$\frac{1}{2}\sum_{k=1}^{K}(t_k - o_k)^2$$

- И нека се користи сигмоидна функција активације и на излазном и на средишњем слоју:

$$o_k = f_{o_k}(net_{o_k}) = \frac{1}{1+e^{-net_{o_k}}}$$

Стохастички градијентни спуст за учење NN

- Тежине се ажурирају на следећи начин:

$$
\begin{aligned}
w_{kj}(t) \quad &+= \quad \Delta w_{kj}(t) + \alpha \Delta w_{kj}(t-1) \\
v_{ji}(t) \quad &+= \quad \Delta v_{ji}(t) + \alpha \Delta v_{ji}(t-1)
\end{aligned}
$$

- Где је α тзв. моменат који дефинише значај претходне промене.

$$
\begin{aligned}
\Delta w_{kj} &= \eta \left(-\frac{\partial E}{\partial w_{kj}} \right) \\
&= -\eta \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial w_{kj}} \\
&= -\eta \delta_{o_k} y_j
\end{aligned}
$$

где је:

$$
\begin{aligned}
\delta_{o_k} &= \frac{\partial E}{\partial net_{o_k}} \\
&= \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial net_{o_k}} \\
&= -(t_k - o_k)(1 - o_k)o_k = -(t_k - o_k) f'_{o_k}
\end{aligned}
$$

и:

$$
\frac{\partial o_k}{\partial net_{o_k}} = \frac{\partial f_{o_k}}{\partial net_{o_k}} = (1 - o_k)o_k = f'_{o_k}
$$

Стохастички градијентни спуст за учење NN (2)

- Слично и за ажурирање тежина између улазног и средњег слоја:

$$\begin{aligned}\Delta v_{ji} &= \eta\left(-\frac{\partial E}{\partial v_{ji}}\right)\\ &= -\eta\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial v_{ji}}\\ &= -\eta\delta_{y_j}z_i\end{aligned}$$

где је:

$$\begin{aligned}\delta_{y_j} &= \frac{\partial E}{\partial net_{y_j}}\\ &= \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial net_{y_j}}\\ &= \sum_{k=1}^{K}\delta_{o_k}w_{kj}f'_{y_j}\end{aligned}$$

и

$$\frac{\partial y_j}{\partial net_{y_j}} = \frac{\partial f_{y_j}}{\partial net_{y_j}} = (1-y_j)y_j = f'_{y_j}$$

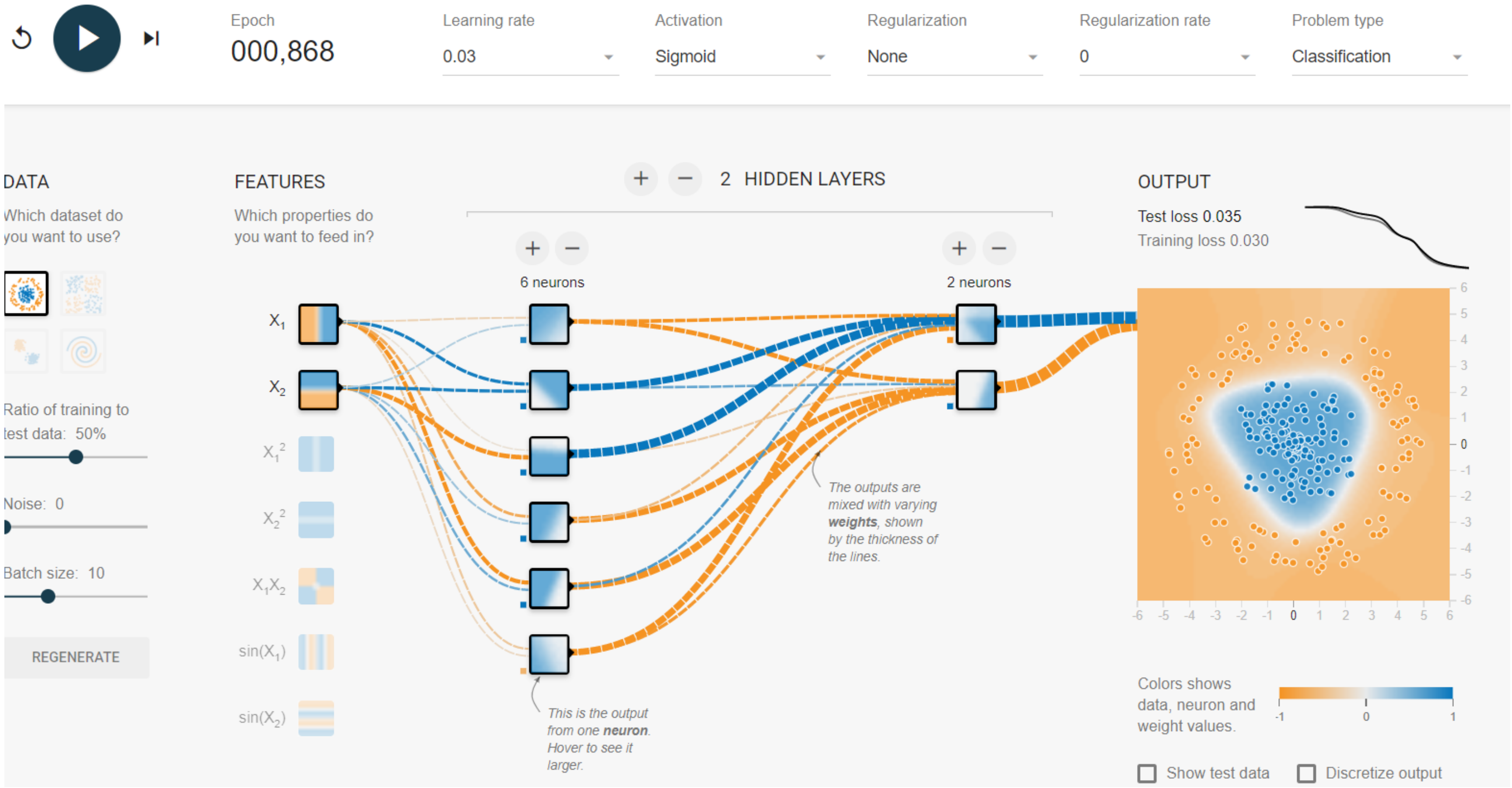
Стохастички градијентни спуст за учење NN (3)

Иницијализуј тежине, η, α, и број епоха $t = 0$;
while *није задовољен услов за завршетак* **do**
    $E = 0$;
    for *сваки тренинг податак p* **do**
        Пропагирај податак унапред и рачунај $y_{j,p}$ ($\forall j = 1, \cdots, J$) и $o_{k,p}$($\forall k = 1, \cdots, K$);
        Рачунај сигнале грешака $\delta_{ok,p}$ и $\delta_{yj,p}$;
        Ажурирај тежине w_{kj} и v_{ji} (пропагација грешака уназад);
        $E += E_p$
    end
    $t = t + 1$
end

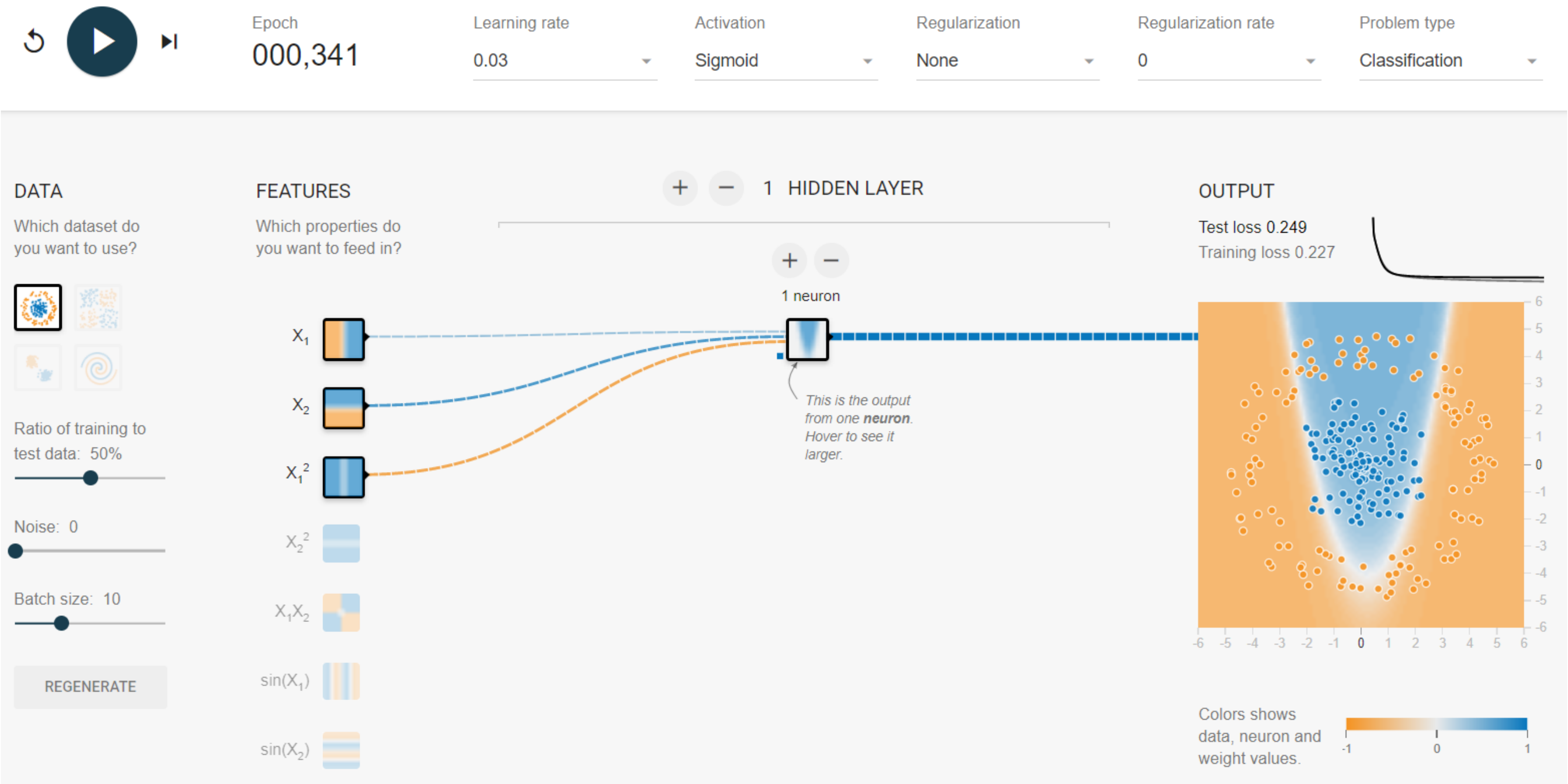
Примери

- На сајту https://playground.tensorflow.org се налази једноставан кориснички интерфејс за тестирање малих мрежа
- Посматрати понашање мреже услед измене:
  - Улазног скупа података
  - Укључених улазних података
  - Удела података за тест и тренинг
  - Броја података
  - Броја слојева (пробати нелинеарни проблем без унутрашњег слоја)
  - Пробати нелинеарни проблем без унутрашњег слоја, али са нелинеарним улазним подацима
  - Броја неурона по слојевима
  - Типа активационе функције
  - Броја епоха
  - Варирати величине тренинг скупа: преприлагођавање, потприлагођавање?
  - Итд.

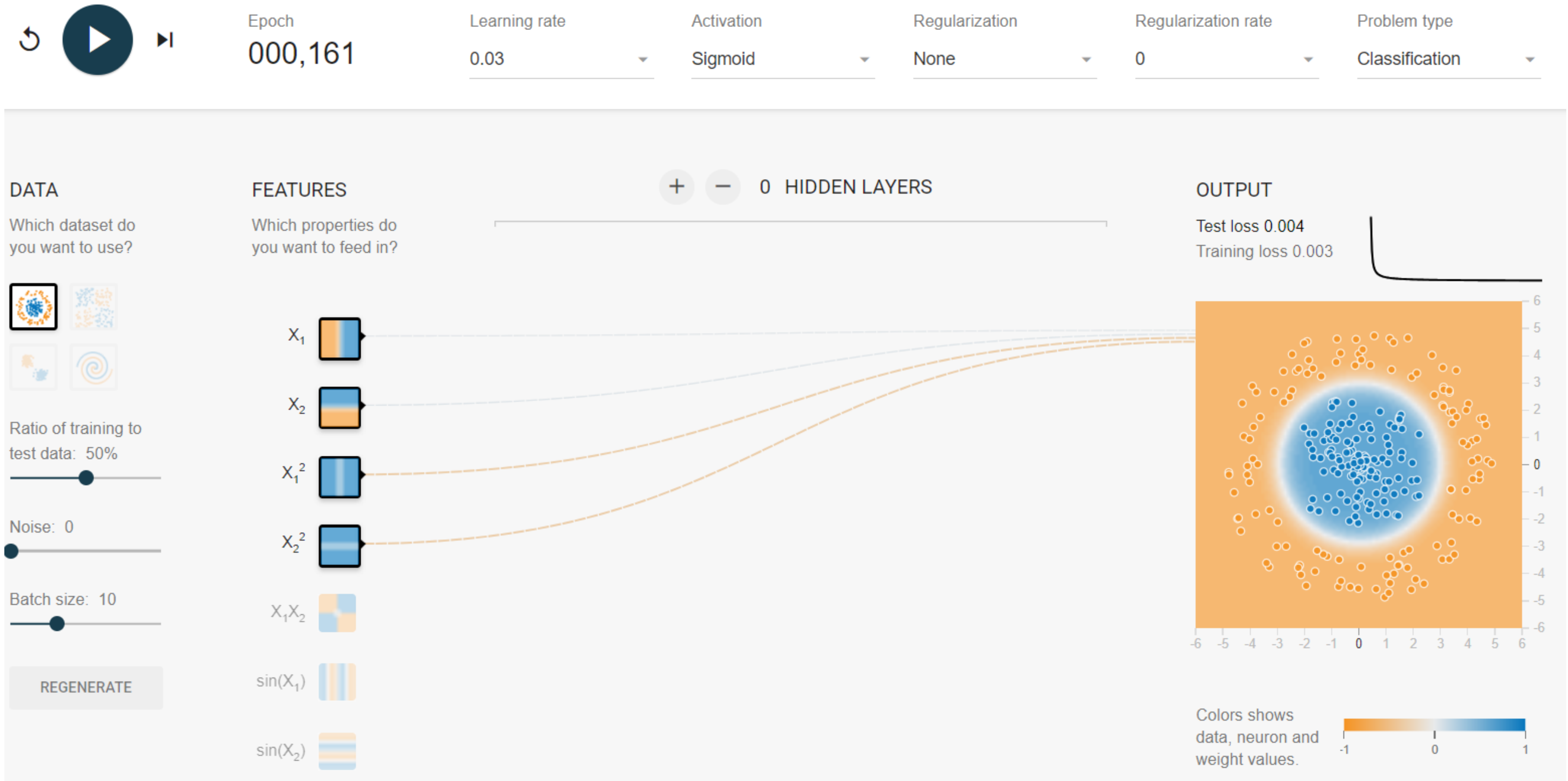
Примери (2)


Epoch
000,868
Learning rate
0.03
Activation
Sigmoid
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
6 neurons
2 neurons
The outputs are mixed with varying **weights**, shown by the thickness of the lines.
This is the output from one **neuron**. Hover to see it larger.
OUTPUT
Test loss 0.035
Training loss 0.030
Colors shows data, neuron and weight values.
Show test data
Discretize output

Примери (3)


Epoch
000,341
Learning rate
0.03
Activation
Sigmoid
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
1 HIDDEN LAYER
1 neuron
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.249
Training loss 0.227
Colors shows data, neuron and weight values.

Примери (4)



Epoch
000,161
Learning rate
0.03
Activation
Sigmoid
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
X_1
X_2
X_1^2
X_2^2
X_1X_2
$\sin(X_1)$
$\sin(X_2)$
0 HIDDEN LAYERS
OUTPUT
Test loss 0.004
Training loss 0.003
Colors shows data, neuron and weight values.

Вештачке неуронске мреже

Ненадгледано учење вештачке неуронске мреже – самоорганизујуће мапе

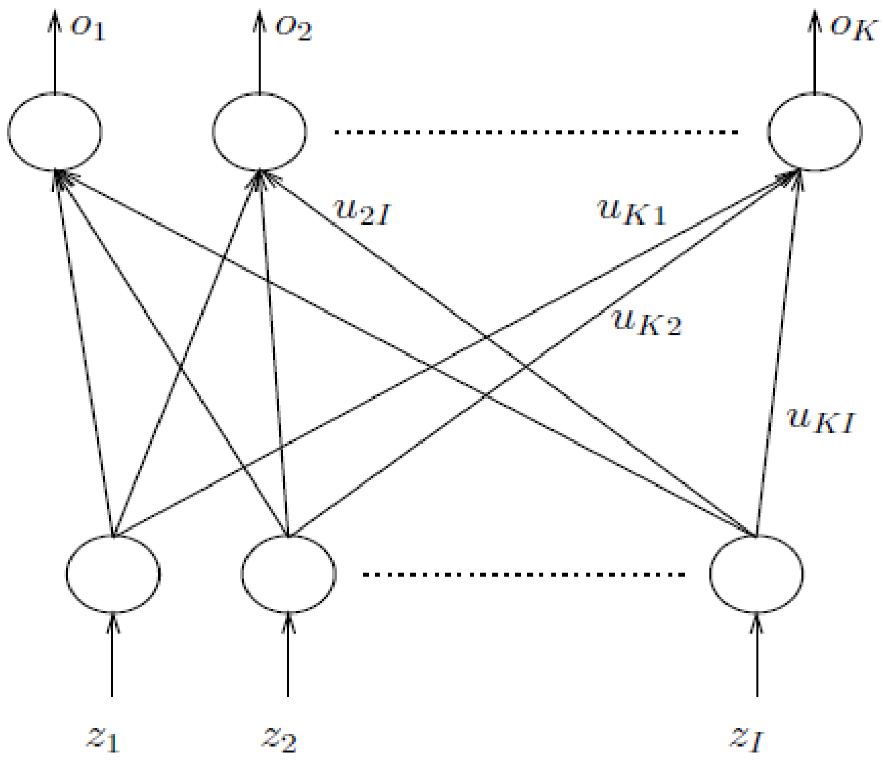
Ненадгледано учење

- Код надгледаног учења, користи се приступ у којем се мрежи даје улаз и очекивани излаз (попут „учитеља“)
  - На основу овога, грешка бива „кажњавана“ ажурирањем тежина у случају да постоји грешка
  - У супротном се не ради ништа
- Код ненадгледаног учења не постоји очекивани излаз
  - Алгоритам учења мора самостално да утврди постојање правилности у улазним подацима
  - Вештачке неуронске мреже омогућавају прављење асоцијација између шаблона (енг. Pattern association)
  - Овакве мреже се још зову и асоцијативна меморија или асоцијативне неуронске мреже
  - Нпр. сећање на слику код човека може да изазове осећај среће, туке, итд.

Асоцијативне неуронске мреже

- Обично двослојне
- Циљ је да омогуће стварање асоцијације – без употребе „учитеља“
- Развој оваквих мрежа заснован на студијама визуелног и звучног кортекста код мозга сисара
- Тополошка организација неурона омогућава асоцијацију
- Додатна пожељна карактеристика је задржавање старих информација и након пристизања нових (надгледаним учењем ово обично не може да се постигне)

Пример асоцијативне мреже



- Функција коју учи оваква мрежа је пресликавање улазног шаблона у излазни

$$f_{NN} : \mathbb{R}^I \rightarrow \mathbb{R}^K$$

Хебово учење

- Названо по неуропсихологу Hebb-у
- Тежине се ажурирају на основу корелације између активационих вредности неурона
- Засновано на хипотези: „потенцијал неурона да испали сигнал је завистан од од потенцијала околних неурона“
- Тежина између два корелисана неурона се појачава

$$u_{ki}(t) = u_{ki}(t-1) + \Delta u_{ki}(t) \qquad \Delta u_{ki}(t) = \eta o_{k,p} z_{i,p}$$

- Измена тежине је већа за оне улазно-излазне парове код којих улазна вредност има јачи ефекат на излазну вредност

Хебово учење (2)

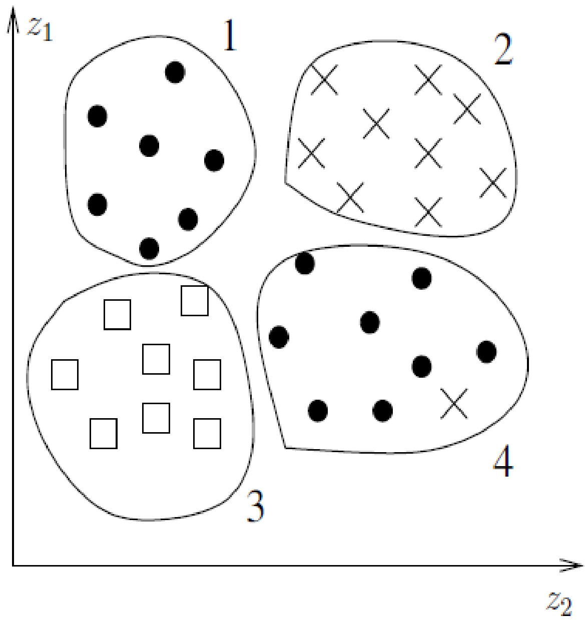
- Проблем је што поновно убацивање улазних шаблона доводи до експоненцијалног раста тежина
- Ово се решава постављањем лимита на вредност тежина
- Пример лимита је нелинеарни фактор заборављања:

$$\Delta u_{ki}(t) = \eta o_{k,p} z_{i,p} - \gamma o_{k,p} u_{ki}(t-1)$$

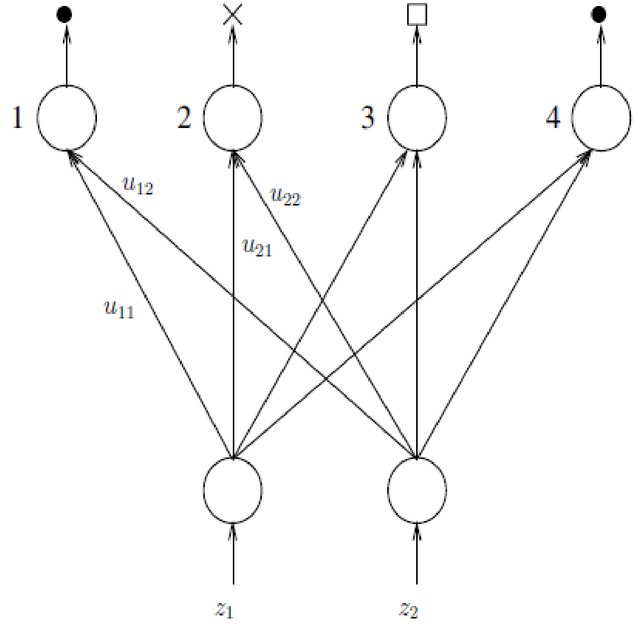
- Где је γ позитивна константа која контролише умањење

LVQ-1 кластеровање

- Енглески назив: Learning Vector Quantizer-1
- Ненадгледана метода учења за кластеровање
- Циљ је скуп од n података груписати у m група:
  - Тако да су елементи из исте групе слични међусобно
  - За меру сличности/различитости се обично користи Еуклидско растојање
  - Излазне вредности (ознаке кластера) се „такмиче“ за улазне податке



(a) Clustering Problem



(b) LVQ-I network

LVQ-1 алгоритам

Иницијализуј тежине мреже, брзину учења и пречник суседства

while *није испуњен услов за завршетак* **do**

for *сваки улазни податак p* **do**

Израчунај Еуклидско растојање, $d_{k,p}$, између улазног вектора $\mathbf{z}_p$ и сваког вектора тежине $\mathbf{u}_k = (u_{k1}, u_{k2}, \cdots, u_{kI})$ по формули:

$$d_{k,p}(\mathbf{z}_p, \mathbf{u}_k) = \sqrt{\sum_{i=1}^{I}(z_{i,p} - u_{ki})^2}$$

Пронађи излазну вредност o_k за коју је растојање $d_{k,p}$ најмање;

Ажурирај све тежине у суседству $\kappa_{k,p}$ формулом: $\Delta u_{ki}(t) = \begin{cases} \eta(t)[z_{i,p} - u_{ki}(t-1)] & \text{if } k \in \kappa_{k,p}(t) \\ 0 & \text{otherwise} \end{cases}$

end

Ажурирај брзину учења;

Смањи пречник суседства;

end

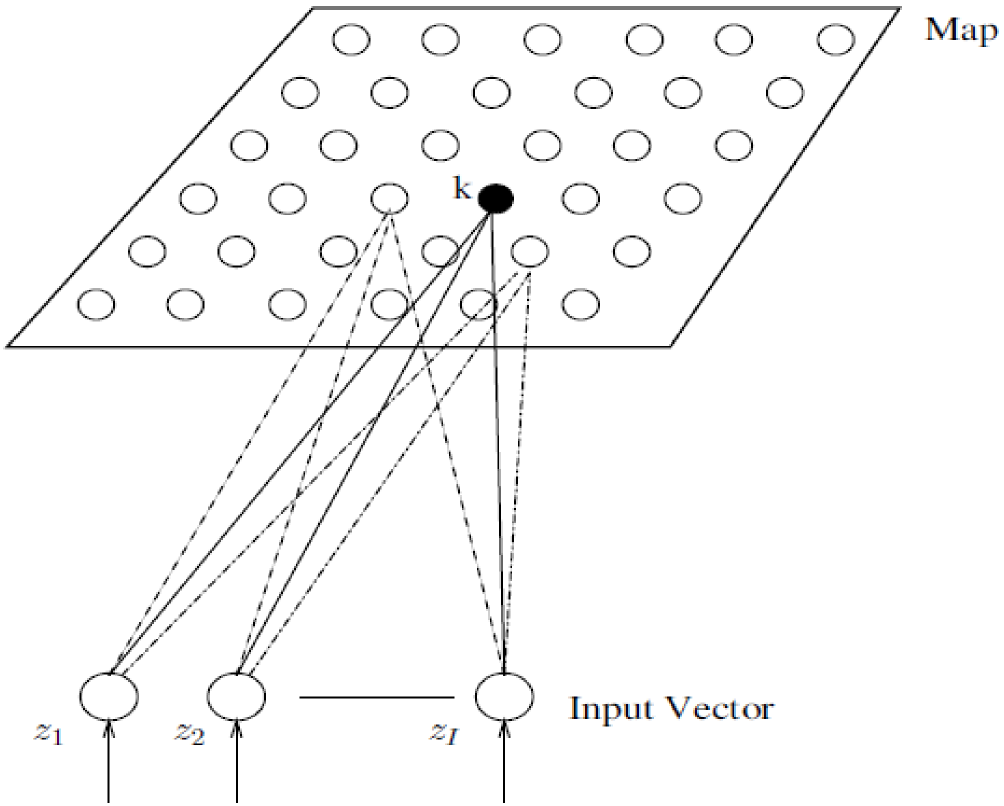
Самоорганизујуће мапе (SOM)

- Енг. Self-organizing feature maps
- Развио их је Кохонен у намери да моделира карактеристике људског целебралног кортекса
- Метода врши пројекцију *I*-димензионог улазног простора у излазни дискретни простор (неки вид компресије)
- Излазни простор је често дводимензиона мрежа вредности
- Идеја је задржавање тополоше структуре улазног простора
  - Ако су два податка близу у улазном простору, биће близу и у излазном
  - Сличне мождане активности активирају блиске неуроне

SOM – стохастичко правило учења

- Засновано на компетитивној стратегији учења
- Врло слично LVQ-1 кластеровању
- Улазни подаци су повезани са одговарајућим неуронима у мапи
  - Мапа је обично квадратног облика
  - Број неурона је мањи од броја тренинг података
  - У идеалном случају број неурона је једнак броју независних тренинг примерака

SOM – стохастичко правило учења (2)



- Вектор тежина за сваки неурон на позицији (k, j) је иницијално насумично подешен:

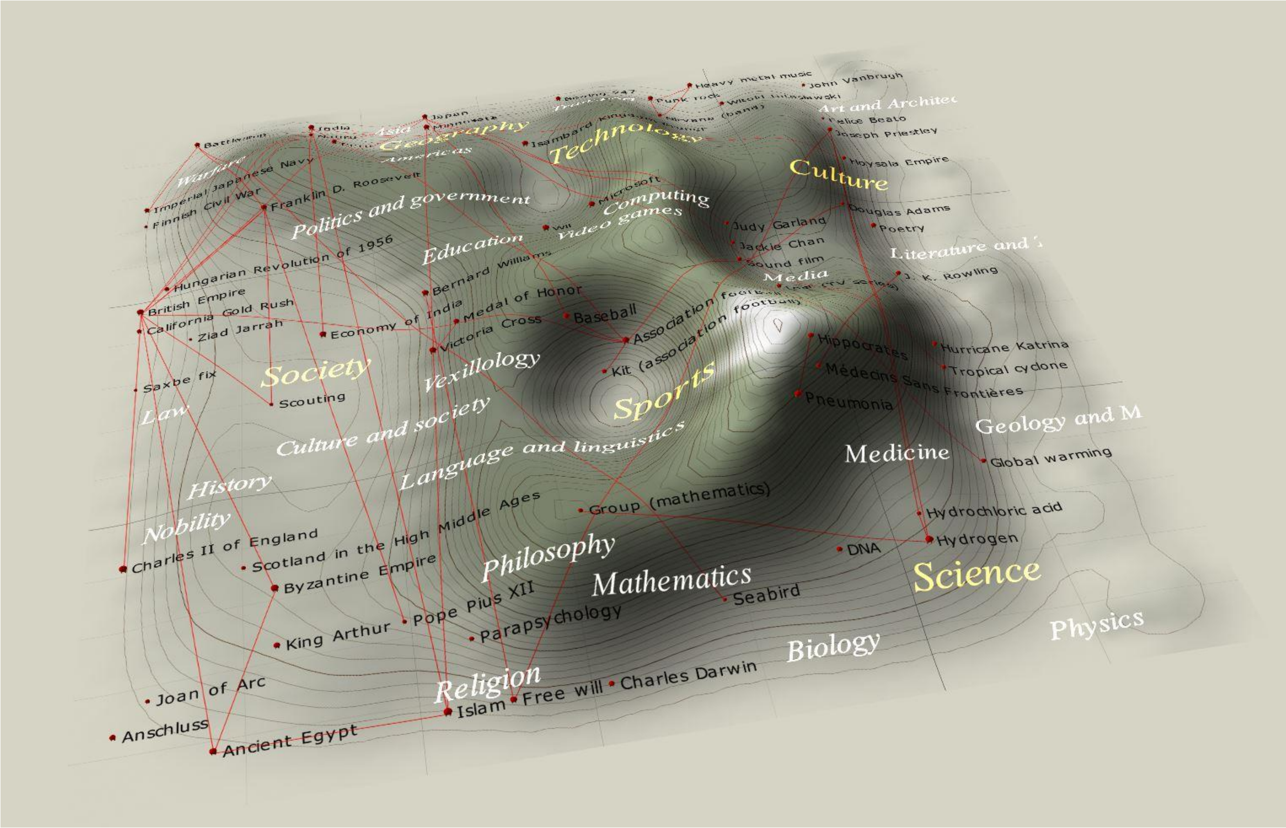
$$\mathbf{w}_{kj} = (w_{kj1}, w_{kj2}, \cdots, w_{KJI})$$

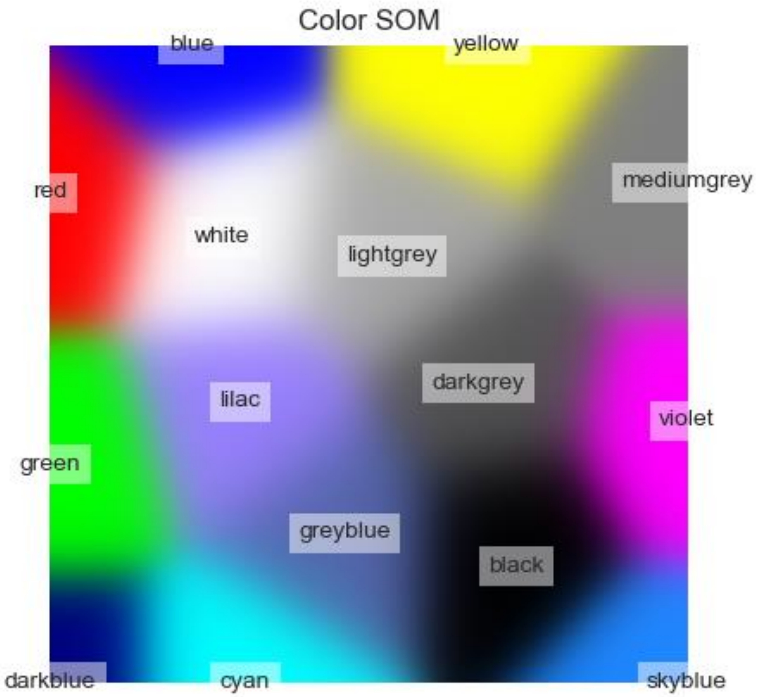
- Сваки улазни податак је повезан са сваким неуроном из мапе
- Приметити да је димензија вектора тежина иста као и димензија улазног податка

SOM – стохастичко правило учења (3)

- За сваки прочитани податак са улазног слоја, проналази се неурон из мапе који има најсличнији тежински вектор
  - Сличност може нпр. бити Еуклидска
- Над тим „победничким“ неуроном врши се корекција тежина у складу са улазним податком
  - Такође се врши корекција тежина суседних неурона пропорицонално њиховој удаљености од „победника“
- Како одмиче тренинг, смањује се и опсег суседних неурона и на самом крају се сматра да неурони више немају суседа

Примери:
распоређивање тема на Википедији и кластеровање боја





Примене SOM

- Анализа слика
- Препознавање звука
- Процесирање сигнала
- Телекомуникације
- Анализа временских серија
- Погодности:
    - Омогућава лаку визуелизацију и интепретацију
    - Области које класификују (категоришу) су видљиве на мапи

Материјали за читање

- https://sci2s.ugr.es/keel/pdf/algorithm/articulo/1990-Kohonen-PIEEE.pdf
- https://cloud.google.com/blog/products/gcp/understanding-neural-networks-with-tensorflow-playground

Алати за развој

- https://www.tensorflow.org/tutorials/
- http://scikit-learn.org/stable/index.html

Задатак 1

- Реализовати неуронску мрежу која је у стању да препознаје ручно написане цифре. Користити скуп података MNIST доступан са адресе http://yann.lecun.com/exdb/mnist/
- Дозвољено је користити и готове алате за неуронске мреже попут оних поменутих на претходном слајду.

Задатак 2

- Имплементирати LVQ-I алгоритам за кластеровање и применити га над Ирис скупом података.
- https://archive.ics.uci.edu/ml/datasets/Iris