Avtokodirniki

Ljupčo Todorovski

Univerza v Ljubljani, Fakulteta za upravo
Institut Jožef Stefan, Odsek za tehnologije znanja (E8)

Maj 2019

Čemu služijo avtokodirniki?

Podatkovna fuzija

- Kombiniranje osnovnih napovednih spremenljivk v nove spremenljivke
- Cilj kombiniranja: brisanje redundantnih in nepomembnih informacij

Vstavitve (*embeddings*) nestrukturiranih podatkov

- Luščenje napovednih spremenljivk iz nestrukturiranih podatkov
- Nestrukturirani podatki: besedila, slike, zvok

Nenadzorovano učenje

Prevod nenadzorovanega učenja v nadzorovano

Pregled vsebine

Definicija avtokodirnikov

- Struktura in usmerjene nevronske mreže
- Vrste avtokodirnikov glede na strukturo
- Taksonomija avtokodirnikov

Osnovni avtokodirnik

- Struktura in aktivacijske funkcije
- Ciljne funkcije in učenje

Napredni avtokodirniki

- Regularizacija
- Robustnost na šumne podatke

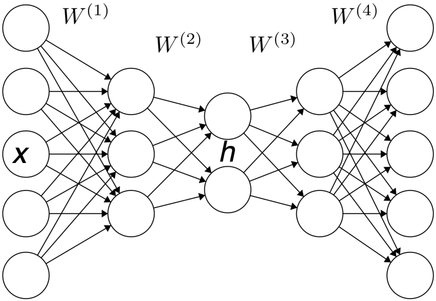
Splošna struktura

$$x \xrightarrow{f} h \xrightarrow{g} \hat{x}$$

- Funkcija f je kodirnik, ki preslika vhod x v kodo h
- Funkcija g je dekodirnik, ki preslika kodo h v $\hat{x}$

- Cilj: rekonstruirati x, t.j., $\min \|x - \hat{x}\|$
- Pri majhni napaki $\|x - \hat{x}\|$, je h dobra vektorska predstavitev x

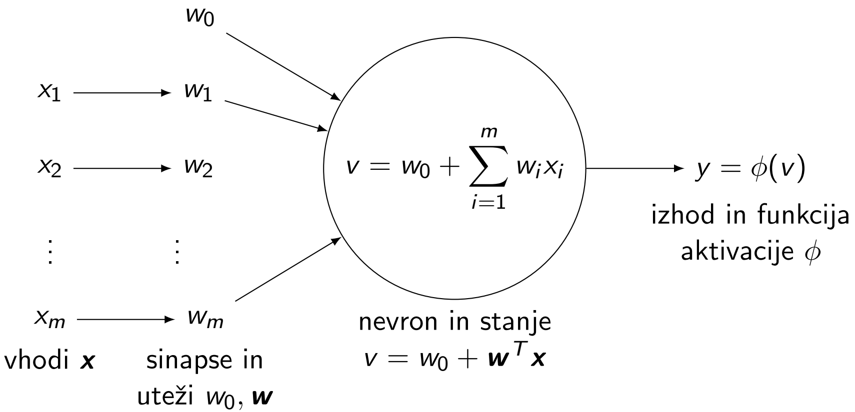
Realizacija z usmerjeno nevronsko mrežo



Kodirnik in dekodirnik implementirani z usmerjeni nevronski mreži

- Sredinski sloj *h* imenujemo *kodirni* sloj (*encoding layer*)
- Strukturi kodirnika in dekodirnika običajno simetrični: enako število skritih slojev (na sliki en) in skritih nevronov (na sliki trije)
- Včasih zahteva po vezanih utežeh (*tied weights*)

Funkcija nevrona



Vektorska notacija: $\boldsymbol{x} = (x_1, x_2, \ldots x_m)^T$ in $\boldsymbol{w} = (w_1, \ldots w_m)^T$

Usmerjene (*feed-forward*) nevronske mreže

Vsaj dva sloja nevronov

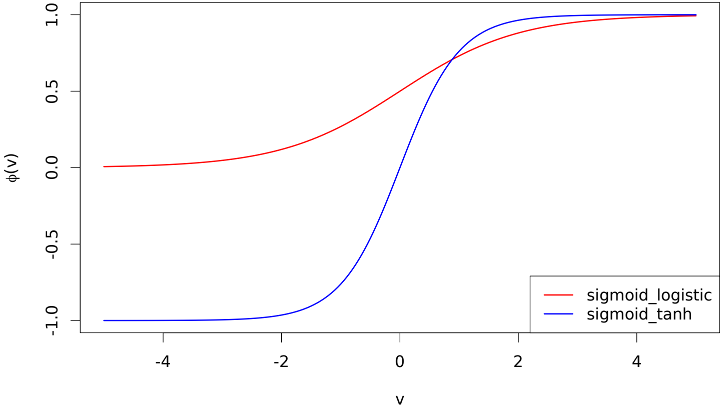
- Sloj vhodnih nevronov, en nevron za vsako vhodno spremenljivko
- Sloj izhodnih nevronov
- Nič ali več slojev skritih (*hidden*) nevronov

Struktura

- Nevroni iz istega sloja niso medsebojno povezani
- Vsaka dva sosednja sloja sta polno povezana: vsak nevron iz sloja $l-1$ je povezan z vsakim nevronom iz sloja l
- Nevroni iz slojev, ki niso sosedni, niso medsebojno povezani

Sigmoidni funkciji aktivacije v usmerjenih NM

$\phi_{logistic}(v) = 1/(1 + exp(-v))$ in $\phi_{tanh}(v) = \tanh v$



Vrste avtokodirnikov glede na strukturo NM

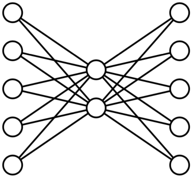
Dimenzija kodirnega sloja $dim(h)$

- $dim(h) < dim(x)$: nepopolni (*undercomplete*) avtokodirniki
- $dim(h) \geq dim(x)$: nad-popolni (*overcomplete*) avtokodirniki

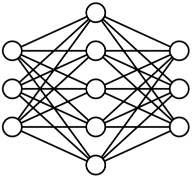
Število skritih slojev

- Kodirnik in dekodirnik brez skritih slojev: plitvi avtokodirniki
- S skritimi sloji: globoki avtokodirniki

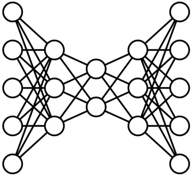
Štiri vrste avtokodirnikov



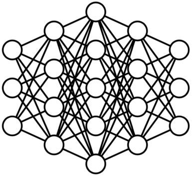
(a) Shallow undercomplete



(b) Shallow overcomplete



(c) Deep undercomplete



(d) Deep overcomplete

Taksonomija avtokodirnikov

Stiskanje in redukcija dimenzije podatkov

Osnovni avtokodirnik

Regularizacija

- Redek (*sparse*) avtokodirnik
- Krčilni (*contractive*) avtokodirnik

Toleranca na šum v podatkih

- Filtrski avtokodirnik
- Robusten avtokodirnik

Plitev nepopoln avtokodirnik

$$\begin{aligned} \boldsymbol{h} = f(\boldsymbol{x}) &= \phi_1(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{w}_0^{(1)}) \\ \hat{\boldsymbol{x}} = g(\boldsymbol{h}) &= \phi_2(\boldsymbol{W}^{(2)}\boldsymbol{h} + \boldsymbol{w}_0^{(2)}) \end{aligned}$$

- Nepopolnost pomeni, da velja $dim(\boldsymbol{h}) < dim(\boldsymbol{x})$
- ϕ_1 in ϕ_2: funkciji aktivacije za kodirnik in dekodirnik
- $\boldsymbol{W}^{(1)}$: matrika uteži kodirnika, dimenzij $dim(\boldsymbol{h}) \times dim(\boldsymbol{x})$
- $\boldsymbol{W}^{(2)}$: matrika uteži dekodirnika, dimenzij $dim(\boldsymbol{x}) \times dim(\boldsymbol{h})$
- $\boldsymbol{w}_0^{(1)}$ in $\boldsymbol{w}_0^{(2)}$: vektorja prostih členov w_0, dimenzij $dim(\boldsymbol{h})$ in $dim(\boldsymbol{x})$
- Spomnite se, da velja $dim(\boldsymbol{x}) = dim(\hat{\boldsymbol{x}})$

Linearen avtokodirnik

$$\begin{aligned}\boldsymbol{h} = f(\boldsymbol{x}) &= \boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{w}_0^{(1)} \\ \hat{\boldsymbol{x}} = g(\boldsymbol{h}) &= \boldsymbol{W}^{(2)}\boldsymbol{h} + \boldsymbol{w}_0^{(2)}\end{aligned}$$

- ϕ_1 in ϕ_2: funkciji aktivacije sta identiteti $\phi_i(x) = x$
- Linearni avtokodirnik opravlja analizo glavnih komponent, PCA
- $\boldsymbol{h}$ vsebuje prvih $k = dim(\boldsymbol{h})$ glavnih komponent učne množice S

Linearni avtokodirnik in PCA sta ekvivalentna

Notacija za izpeljavo

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T,\ \hat{\boldsymbol{X}} = (\boldsymbol{U}_{\cdot,\leq k}\boldsymbol{\Sigma}_{\leq k,\leq k})\,\boldsymbol{X}$$

- $\boldsymbol{X}$ je matrika s stolpci, ki ustrezajo učnim primerom iz S
- $\hat{\boldsymbol{X}}$ je matrika s stolpci, ki ustrezajo učnim primerom iz S, kot jih rekonstruira avtokodirnik
- Torej je matrika uteži dekodirnika $\boldsymbol{W}^{(2)} = \boldsymbol{U}_{\cdot,\leq k}\Sigma_{\leq k,\leq k}$
- Prvih $k = dim(\boldsymbol{h})$ glavnih komponent: $\boldsymbol{h}_{\boldsymbol{X}} = \boldsymbol{V}^T_{\cdot,\leq k}$

Dokazati moramo, da je tudi $\boldsymbol{h}_{\boldsymbol{X}} = \boldsymbol{W}^{(1)}\boldsymbol{X}$ (na tabli)

Zakaj rabimo še eno različico PCA?

Ker lahko s pomočjo funkcij aktivacije in NM posplošimo na *nelinearno* analizo glavnih komponent

Ciljne funkcije za avtokodirnik

$$E(\boldsymbol{W}, \boldsymbol{w}; S) = \sum_{\boldsymbol{x} \in S} L(\boldsymbol{x}, \hat{\boldsymbol{x}}),\ \hat{\boldsymbol{x}} = g(f(\boldsymbol{x}))$$

Funkcija izgube L_{SE} za numerične spremenljivke, kvadratna napaka

$$L_{SE}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{2} \sum_{i=1}^{p} (x_i - \hat{x}_i)^2$$

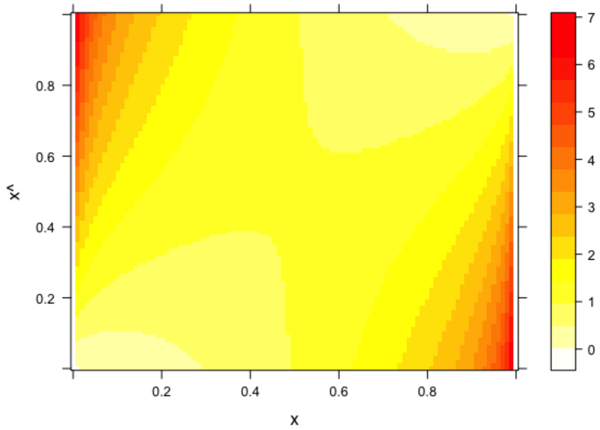
Dekodirnik lahko uporablja linearno ali sigmoidno funkcijo aktivacije.

Funkcija izgube L_{CE} za Boolove spremenljivke, prečna entropija

$$L_{CE}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \sum_{i=1}^{p} (x_i \log \hat{x}_i + (1 - x_i) \log(1 - \hat{x}_i))$$

Dekodirnik uporablja sigmoidno (logistično) funkcijo aktivacije.

Prečna entropija



Izračun gradienta: raven izhodnih nevronov

$$E = \frac{1}{2}\sum_i (x_i - \hat{x}_i)^2,\ \hat{x}_i = y_i^{(L+1)} = \phi(v_i^{(L+1)}),\ v_i^{(L+1)} = \sum_j w_{ji}^{(L+1)} y_j^{(L)}$$

$$\begin{aligned}
\frac{\partial E}{\partial w_{ji}^{(L+1)}} &= \frac{\partial E}{\partial \hat{x}_i} \frac{\partial \hat{y}_i^{(L+1)}}{\partial v_i^{(L+1)}} \frac{\partial v_i^{(L+1)}}{\partial w_{ji}^{(L+1)}} \\
&= -(x_i - \hat{x}_i)\, \phi'(v_i^{(L+1)})\, \frac{\partial \sum_k w_{ki}^{(L+1)} y_k^{(L)}}{\partial w_{ji}^{(L+1)}} \\
&= (\hat{x}_i - x_i)\, \phi'(v_i^{(L+1)})\, y_j^{(L)}
\end{aligned}$$

$$\Delta w_{ji}^{(L+1)} = -\eta\,(\hat{x}_i - x_i)\,\phi'(v_i^{(L+1)})\, y_j^{(L)},\ \frac{\partial E}{\partial y_i^{(L+1)}} = \hat{x}_i - x_i$$

Vzvratno razširjanje napake: ravni skritih nevronov

$$\frac{\partial E}{\partial w_{ji}^{(l)}} = \frac{\partial E}{\partial y_i^{(l)}} \frac{\partial y_i^{(l)}}{\partial v_i^{(l)}} \frac{\partial v_i^{(l)}}{\partial w_{ji}^{(l)}} = \frac{\partial E}{\partial y_i^{(l)}} \phi'(v_i^{(l)})\, y_j^{(l-1)}$$

Trik za izračun $\frac{\partial E}{\partial y_i^{(l)}}$ iz $\frac{\partial E}{\partial y_k^{(l+1)}}$

$$\begin{aligned}\frac{\partial E}{\partial y_i^{(l)}} &= \sum_k \frac{\partial E}{\partial y_k^{(l+1)}} \frac{\partial y_k^{(l+1)}}{\partial v_k^{(l+1)}} \frac{\partial v_k^{(l+1)}}{\partial y_i^{(l)}} \\ &= \sum_k \frac{\partial E}{\partial y_k^{(l+1)}} \phi'(v_k^{(l+1)})\, w_{ik}^{(l+1)}\end{aligned}$$

Vzvratno razširjanje napake: iteracija

$$\Delta w_{ji}^{(l)} = -\eta\, \phi'(v_i^{(l)})\, y_j^{(l-1)} \sum_k \frac{\partial E}{\partial y_k^{(l+1)}}\, \phi'(v_k^{(l+1)})\, w_{ik}^{(l+1)}$$

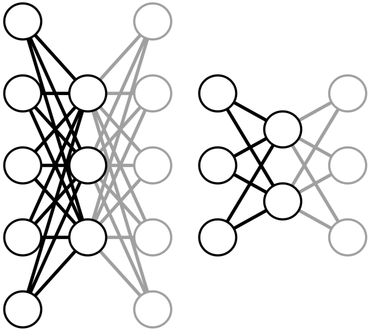
- Iteracija pravila od ravni L do ravni 1
- Na ravni 1 upoštevamo $y_i = x_i$

Kopičenje

Trik za določanje začetnih vrednosti uteži

- Uspeh gradientne metode odvisen od začetnih vrednosti (sreče)
- Zato avtokodirniki uporabljajo kopičenje
- Kopičenje deluje tako, da optimiziramo uteži po slojih
- Tako optimizirane uteži uporabimo kot začetne vrednosti za učenje uteži celotne NM naenkrat

Kopičenje: optimizacija uteži za prva dva sloja



Vizualizacija

Za vsak nevron i v kodirnem sloju

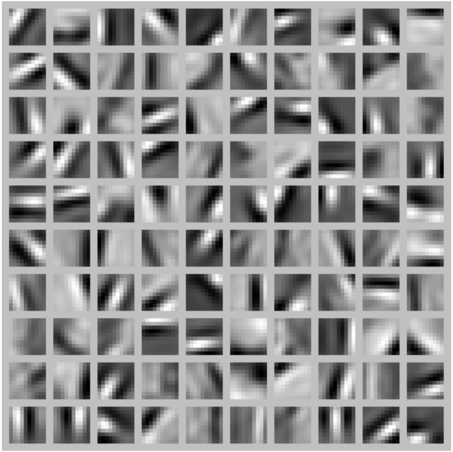
Sestavimo umeten primer $\boldsymbol{x}$ za katerega velja

1. Da je norma primera $\|\boldsymbol{x}\|_2 = \sum_{i=1}^{p} x_i \leq 1$
2. Da ima nevron i maksimalno vrednost stanja v_i

V primeru kodirnika brez skritih slojev je $x_i = w_{ji} / \sqrt{\sum_{j=1}^{p} w_{ji}^2}$

Tako za vsako dimenzijo kode h dobimo občutek o tem kaj dela.

Primer vizualizacije za učno množico slik



Zakaj osnovni avtokodirnik slabo dela?

Nepopoln avtokodirnik

- Kodira (stisne) primere iz učne množice
- Tako si, na primer, lahko zapomni le indeks primera
- Iz indeksa dekodirnik, če je dovolj globok, rekonstruira primer

Nad-popoln avtokodirnik

- Omogoča preprileganje učnim podatkom
- Lahko si v kodi zapomni cele primere brez izgube
- Ne dobimo nič pametnega, ni posploševanja

Regularizacija

$$E(\boldsymbol{W}, \boldsymbol{w}; S) = \sum_{\boldsymbol{x} \in S} L(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \lambda \, \Omega(\boldsymbol{W}, \boldsymbol{w}; S)$$

Ω je kazen za "slabo obnašanje" avtokodirnika

- Kazen za prevelike vrednosti uteži: redek avtokodirnik
- Kazen za preveliko občutljivost na spremembe primerov: krčilni AK

Redčenje uteži

$$\Omega(\boldsymbol{W}, \boldsymbol{w}; S) = \|\boldsymbol{W}\|_F^2$$

$\|\boldsymbol{W}\|_F$ je Forbeniusova norma matrike $\boldsymbol{W}$

Naivni pristop k regularizaciji

- Povzroča redčenje (zmanjševanje) uteži, ne pa redčenje kode
- Sicer zelo enostaven za implementacijo
- Odvodom $\partial E / \partial w_{ij}$ prištejemo $2w_{ij}$

Redčenje kode

$$\Omega(\boldsymbol{W}, \boldsymbol{w}; S) = \sum_{i=1}^{k=dim(\boldsymbol{h})} KL(\rho \| \hat{\rho}_i) = \sum_{i=1}^{k} (\rho \log \frac{\rho}{\hat{\rho}_i} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_i})$$

KL je Kullback-Leibler divergenca, ki meri razlike med porazdelitvami

Temelji na povprečni aktivaciji nevrona v kodirnem sloju

$$\hat{\rho}_i = \frac{1}{|S|} \sum_{\boldsymbol{x} \in S} f_i(\boldsymbol{x})$$

- Aktivacija se zgodi, če je izhod npr. večji od 0.5
- Aktivacio nato modeliramo kot Bernoullijevo slučajno spremenljivko
- Za pričakovano vrednost povprečja ρ nastavimo nizko vrednost
- $\rho = 0.1$: zahtevamo, da bo le 10% kodirnih nevronov aktivnih

Kazen za preveliko občutljivost

$$\Omega(\boldsymbol{W}, \boldsymbol{w}; S) = \sum \sum_{\boldsymbol{x} \in S} \|J_f(\boldsymbol{x})\|_F^2$$

Občutljivost merimo kot normo Jacobijeve matrike

- $J_f[i,j] = \partial f_i / \partial x_j$: odvod kodirnika po vhodu
- Odvod i-te komponente kodirnika po j-ti komponenti vhoda

- Kontradikcija: mi želimo, da bi bil kodirnik občutljiv
- Ilustracija na tabli

Osnovna ideja

$$x \xrightarrow{q(\tilde{x}|x)} \tilde{x} \xrightarrow{f} h \xrightarrow{g} \hat{x}$$

Umetno šumenje podatkov $\tilde{x} \sim q(\tilde{x}|x)$

- Za numerične spremenljivke je lahko Gaussov šum $N(x_i, \sigma)$
- Za Boolove pa vrednost x_i zamenjamo z 0 z verjetnostjo q

- Še vedno želimo rekonstruirati prave vrednosti x
- Zveni hecno: zakaj deluje - na tabli?

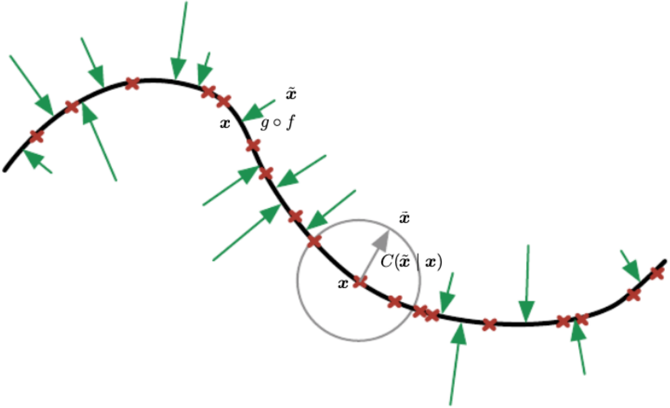
Ciljna funkcija

$$E(\boldsymbol{W}, \boldsymbol{w}; S) = \sum_{\boldsymbol{x} \in S} \mathbb{E}_{\tilde{x} \sim \boldsymbol{q}(\tilde{x}|x)}[L(\boldsymbol{x}, \hat{\boldsymbol{x}})]$$

Pozor: $\hat{x} = g(f(\tilde{x}))$ in ne $\hat{x} = g(f(x))$!

Ker je šum stohastičen proces, moramo računati pričakovano vrednost funkcije izgube.

Grafična ponazoritev delovanja



Sprememba funkcije izgube

$$L_{correntropy}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \sum_{i=1}^{p} \mathcal{K}_\sigma(x_i - \hat{x}_i),\ \mathcal{K}_\sigma(\alpha) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\alpha^2}{2\,\sigma^2}\right)$$

$\mathcal{K}_\sigma$ je jedrna funkcija, ki blaži vpliv osamelcev na vrednost ciljne funkcije.

Viri in implementacije

Viri

- Neverjetno dobra vadnica (Charte in ost. 2018)
- Deli poglavja 14 učbenika (Goodfellow in ost. 2016) *Deep Learning*

Implementacije

- Običajna okolja za globoke NM
- Paketa za R *Autoencoder* in (v CRANu nepodprti) *SAENET*