Istraživanje podataka

Vežbe 9

16. April 2021

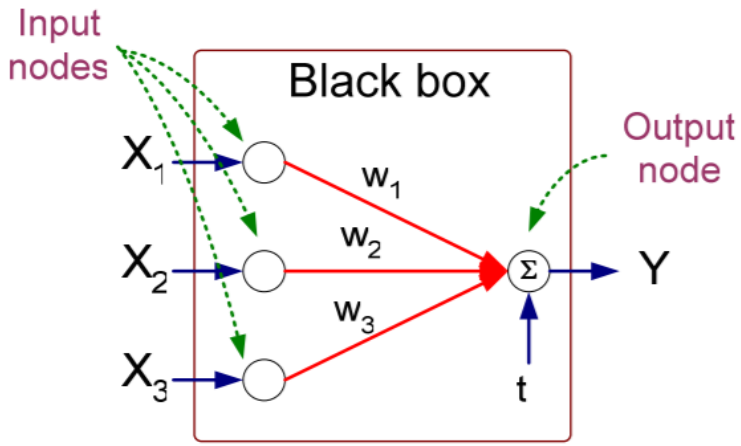
Outline

1. Veštačke neuronske mreže
2. PCA - Analiza glavnih komponenti (Principal Component Analysis)
3. Zadatak za samostalan rad

Outline

1. Veštačke neuronske mreže
2. PCA - Analiza glavnih komponenti (Principal Component Analysis)
3. Zadatak za samostalan rad

Perceptron - neuronska mreža bez skrivenih slojeva



Model Perceptrona

$$Y = I(\sum_i w_i X_i - t) \quad \text{or}$$

$$Y = sign(\sum_i w_i X_i - t)$$

Perceptron

Učenje modela

$$w_j^{(k+1)} = w_j^{(k)} + \eta(y_i - \hat{y}_i^{(k)})x_{ij}$$

Za

- $y = +1$ i $\hat{y} = -1$ važi $(y - \hat{y}) = 2$
- $y = -1$ i $\hat{y} = +1$ važi $(y - \hat{y}) = -2$

- za linearno razdvojive probleme klasifikacije

Perceptron - zadatak

Date su instance klase 1: $\{\langle 0,0\rangle, \langle 1,0\rangle\}$ i instance klase -1: $\{\langle 0,1\rangle, \langle 1,1\rangle\}$. Ako su inicijalne težine $w = \langle 0,0,0\rangle$, a $\eta = 0,5$ trenirati perceptron koristeći instance u seldećem redosledu $\{\langle 0,0\rangle, \langle 1,1\rangle, \langle 0,1\rangle, \langle 1,0\rangle\}$.

Perceptron - zadatak

Prva iteracija treninga u kojoj za trening koristimo instancu $\langle 0, 0 \rangle$ daje:

$$w_0 \leftarrow 0 + 0,5 * (1 - sgn(\langle 0, 0, 0 \rangle * \langle 1, 0, 0 \rangle)) * 1 = 0$$

$$w_1 \leftarrow 0 + 0,5 * (1 - sgn(\langle 0, 0, 0 \rangle * \langle 1, 0, 0 \rangle)) * 0 = 0$$

$$w_2 \leftarrow 0 + 0,5 * (1 - sgn(\langle 0, 0, 0 \rangle * \langle 1, 0, 0 \rangle)) * 0 = 0$$

Perceptron - zadatak

U drugoj iteraciji koristimo instancu : $\langle 1, 1 \rangle$ i $w = \langle 0, 0, 0 \rangle$:

$$w_0 \leftarrow 0 + 0,5 * (-1 - sgn(\langle 0, 0, 0 \rangle * \langle 1, 1, 1 \rangle)) * 1 = -1$$

$$w_1 \leftarrow 0 + 0,5 * (-1 - sgn(\langle 0, 0, 0 \rangle * \langle 1, 1, 1 \rangle)) * 1 = -1$$

$$w_2 \leftarrow 0 + 0,5 * (-1 - sgn(\langle 0, 0, 0 \rangle * \langle 1, 1, 1 \rangle)) * 1 = -1$$

Perceptron - zadatak

U trećojc iteraciji koristimo instancu : $\langle 0, 1 \rangle$ i $w = \langle -1, -1, -1 \rangle$:

$$w_0 \leftarrow -1 + 0,5 * (-1 - sgn(\langle -1, -1, -1 \rangle * \langle 1, 0, 1 \rangle)) * 1 = -1$$

$$w_1 \leftarrow -1 + 0,5 * (-1 - sgn(\langle -1, -1, -1 \rangle * \langle 1, 0, 1 \rangle)) * 0 = -1$$

$$w_2 \leftarrow -1 + 0,5 * (-1 - sgn(\langle -1, -1, -1 \rangle * \langle 1, 0, 1 \rangle)) * 1 = -1$$

Perceptron - zadatak

U četvrtoj iteraciji koristimo instancu : $\langle 1, 0\rangle$ i $w = \langle -1, -1, -1\rangle$:

$$w_0 \leftarrow -1 + 0,5 * (1 - sgn(\langle -1, -1, -1\rangle * \langle 1, 1, 0\rangle)) * 1 = 0$$

$$w_1 \leftarrow -1 + 0,5 * (1 - sgn(\langle -1, -1, -1\rangle * \langle 1, 1, 0\rangle)) * 1 = 0$$

$$w_2 \leftarrow -1 + 0,5 * (1 - sgn(\langle -1, -1, -1\rangle * \langle 1, 1, 0\rangle)) * 0 = -1$$

Perceptron - zadatak

$$\hat{y}_1 = sgn(\langle 0, 0, -1\rangle * \langle 1, 0, 0\rangle) = 1, y_1 = 1$$

$$\hat{y}_2 = sgn(\langle 0, 0, -1\rangle * \langle 1, 1, 0\rangle) = 1, y_2 = 1$$

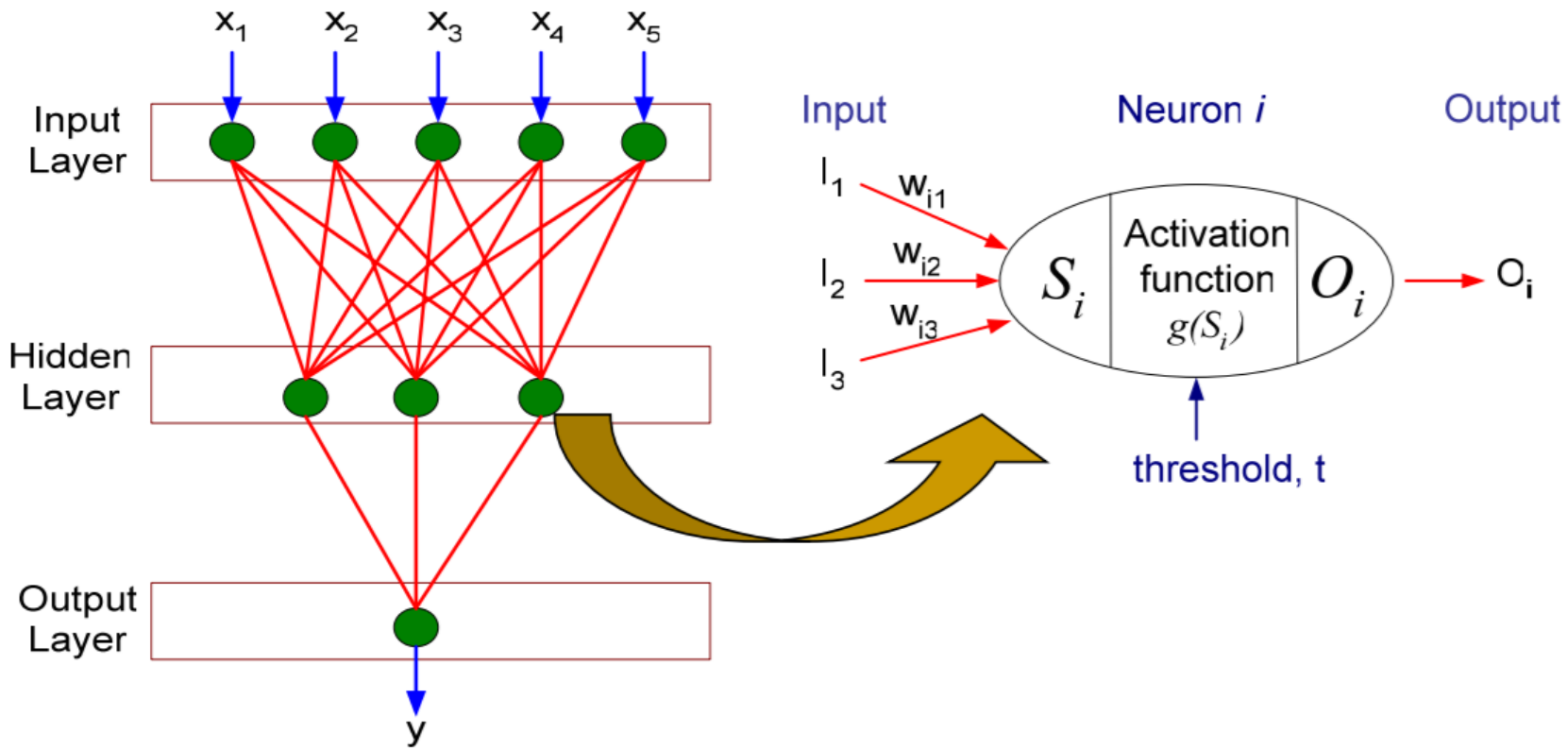
$$\hat{y}_3 = sgn(\langle 0, 0, -1\rangle * \langle 1, 0, 1\rangle) = -1, y_3 = -1$$

$$\hat{y}_4 = sgn(\langle 0, 0, -1\rangle * \langle 1, 1, 1\rangle) = -1, y_4 = -1$$

Perceptron - zadatak

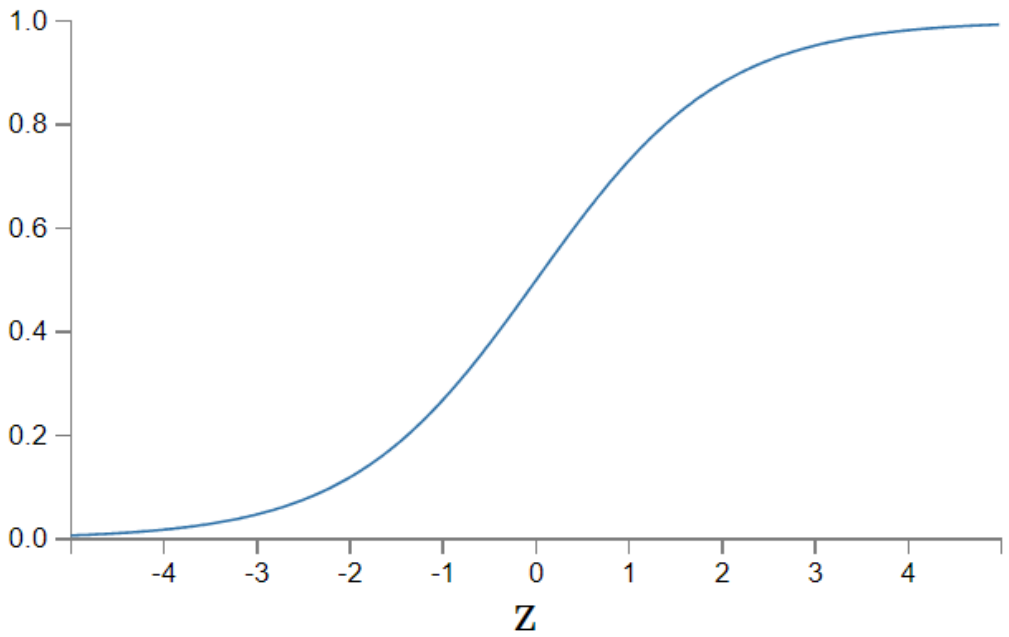
Korišćenjem dobijenog vektora $w = \langle 0, 0, -1 \rangle$, dobijamo tačnu klasifikaciju za svaku instancu, tako da više ne može doći do njegove promene i algoritam se zaustavlja.

Neuronska mreža sa skrivenim slojevima

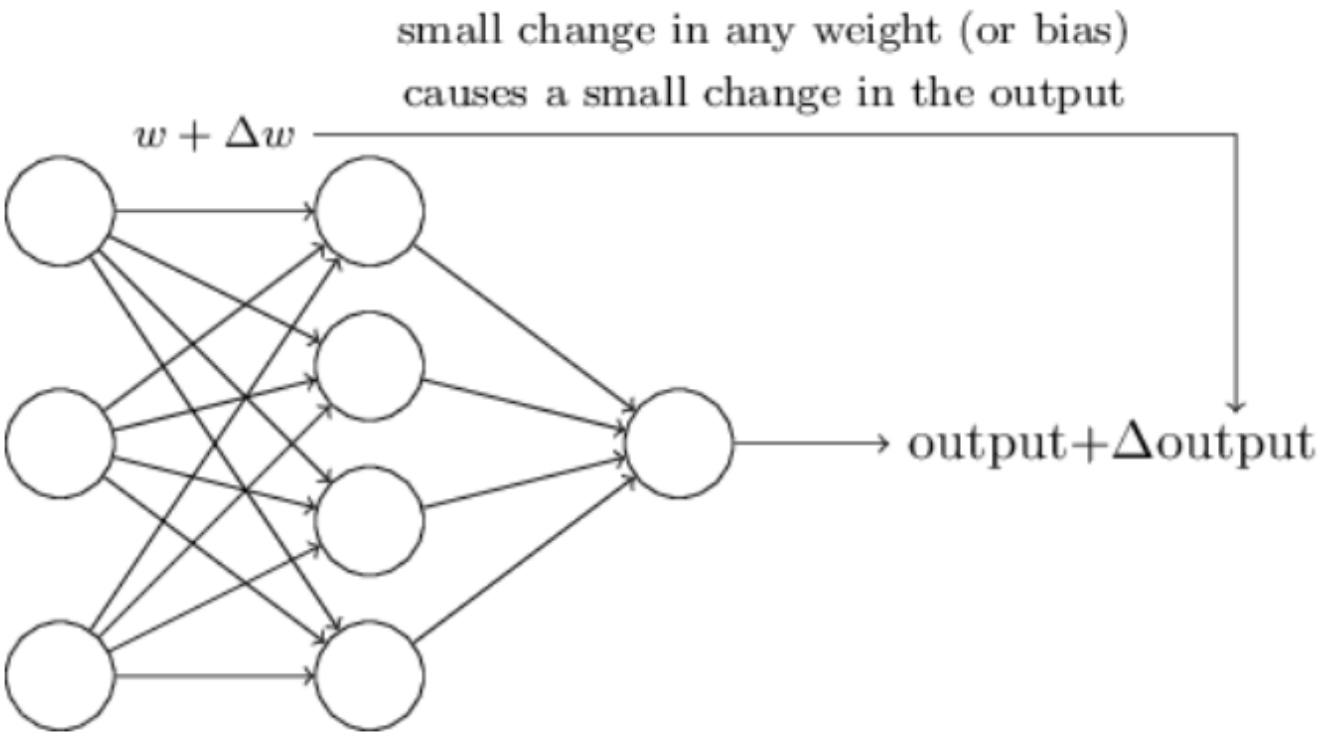


Neuronska mreža sa skrivenim slojevima

Koriste se i druge aktivcione funkcije (osim *sign*). Npr. pozadinska (eng. logistic/sigmoid) funkcija $\frac{1}{1+e^{-z}}$ kod koje mala promena u težinama dovodi do male promene u izlazu (rezultatu aktivacione funkcije).



Neuronska mreža sa skrivenim slojevima

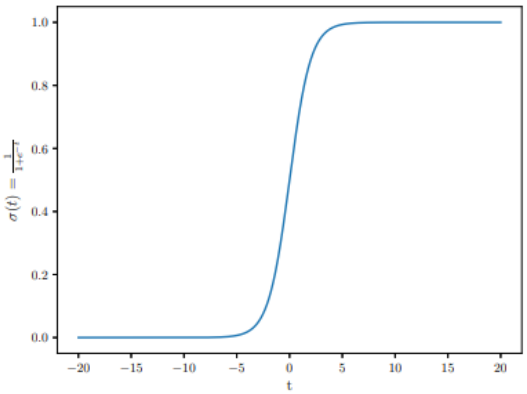


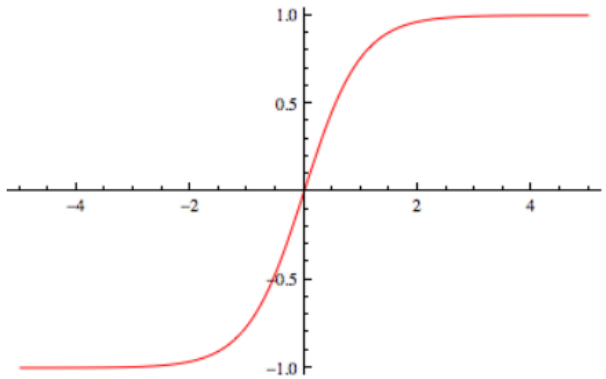
Neuronska mreža sa skrivenim slojevima

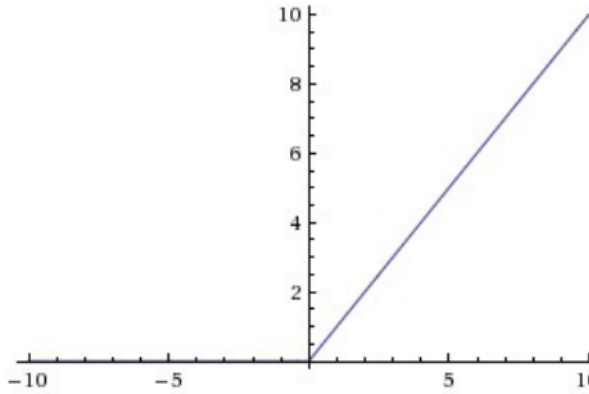
Najčeš' ce f-je:

- Identity
$$f(x) = x$$
- Sigmoidna
$$f(x) = 1/(1 + e^{-x})$$
- Tangens hiperbolički (tanh)
$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$
- Ispravljačka linearna funkcija (relu)
$$f(x) = max(0, x)$$

Neuronska mreža sa skrivenim slojevima







Pristup gradijentni spust

Gradijentni spust je algoritam za nalaženje lokalnog minimuma funkcije.

Cilj - minimizacija zbira kvadrata greške

$$E(w) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Učenje modela

$$w_j = w_j - \eta * \frac{\partial E(w)}{\partial w_j}$$

Pristup gradijentni spust

Algoritam propagacije unatrag (eng. backpropagation).
Faze:

- *unapred* - težine iz prethodne iteracije se koriste za računanje izlazne vrednosti svakog neuronu
- *unazad* - formula za ažuriranje težina se primenjuje u obrnutom smeru

Ciljni atribut sa više klasa

- jedan izlazni neuron za svaku klasu, bira se klasa čiji neuron da najveći izlaz
- funkcija aktivacije izlaznih neurona - funkcija mekog maksimuma (eng. softmax)
  $softmax(x) = (\frac{e^{x1}}{\sum_{i=1}^{C} e^{x_i}}, \ldots, \frac{e^{xC}}{\sum_{i=1}^{C} e^{x_i}})$
  gde je x_i izlaz neurona vezanog za klasu i.
- funkcija gubitka - unakrsna entropija

Neuronska mreža sa skrivenim slojevima u biblioteci scikit-learn

- *sklearn.neural_network.MLPClassifier*
- parametri
  - *hidden_layer_sizes* -broj neurona po skrivenim slojevima: default (100,)
  - aktivaciona funkcija
    - *identity* : f(x) = x
    - *logistic*: sigmoidna funkcija, f(x) = 1 / (1 + exp(-x))
    - *tanh*: tangens hiperbolički funkcija, f(x) = tanh(x)
    - *relu*: ispravljačka linearna jedinica, f(x) = max(0, x)

Neuronska mreža sa skrivenim slojevima u biblioteci scikit-learn

- *solver* - rešavač za optimizaciju težina
  - *sgd* : stohastički gradijentni spust
- *max_iter* - maksimalan broj iteracija, default 200
- *learning_rate_init* - inicijalna stopa učenja, default 0.001

Neuronska mreža sa skrivenim slojevima u biblioteci scikit-learn

- *learning_rate* - stopa učenja pri ažuriranju težina
  - *constant* : konstantna, zadata sa *learning_rate_init*
  - *invscaling*:postepeno smanjenje stope ucenja u koraku t, *effective_learning_rate = learning_rate_init / pow(t, power_t)*
  - *adaptive*: stopa učenja se ne menja dok se vrednost fje gubitka smanjuje. Kad se u dva uzastopna ciklusa gubitak ne smanji za bar vrednost tol, trenutna stopa učenja se deli sa 5.

Neuronska mreža sa skrivenim slojevima u biblioteci scikit-learn

- *power_t* - eksponent za brzinu učenja obrnutog skaliranja, default 0.5
- *tol* - tolerancija optimizacije za gubitak, default 1e-4
- *shuffle* - da li izvršiti mešanje instanci za svaku iteraciju, default=False
- *early_stopping* - da li izvršiti rano zaustavljanje kada se preciznost nad skupom za validaciju ne povećava default=False
- *validation_fraction* - koji deo skupa za treniranje se koristi za validaciju. Primenjivo ako je *early_stopping*=True, default 0.1

Neuronska mreža sa skrivenim slojevima u biblioteci scikit-learn

- atributi
  - *coefs_* - i. element u listi predstavlja matricu težina koja odgovara sloju i+1.
  - *intercepts_* - otkloni
  - *n_iter_* - broj izvršenih iteracija
  - *n_layers_* - broj slojeva

Neuronska mreža sa skrivenim slojevima u biblioteci scikit-learn

- metode
  - *fit(X, y)* - za treniranje modela
  - *predict(X)* - za predviđanje klasa

Outline

1. Veštačke neuronske mreže
2. PCA - Analiza glavnih komponenti (Principal Component Analysis)
3. Zadatak za samostalan rad

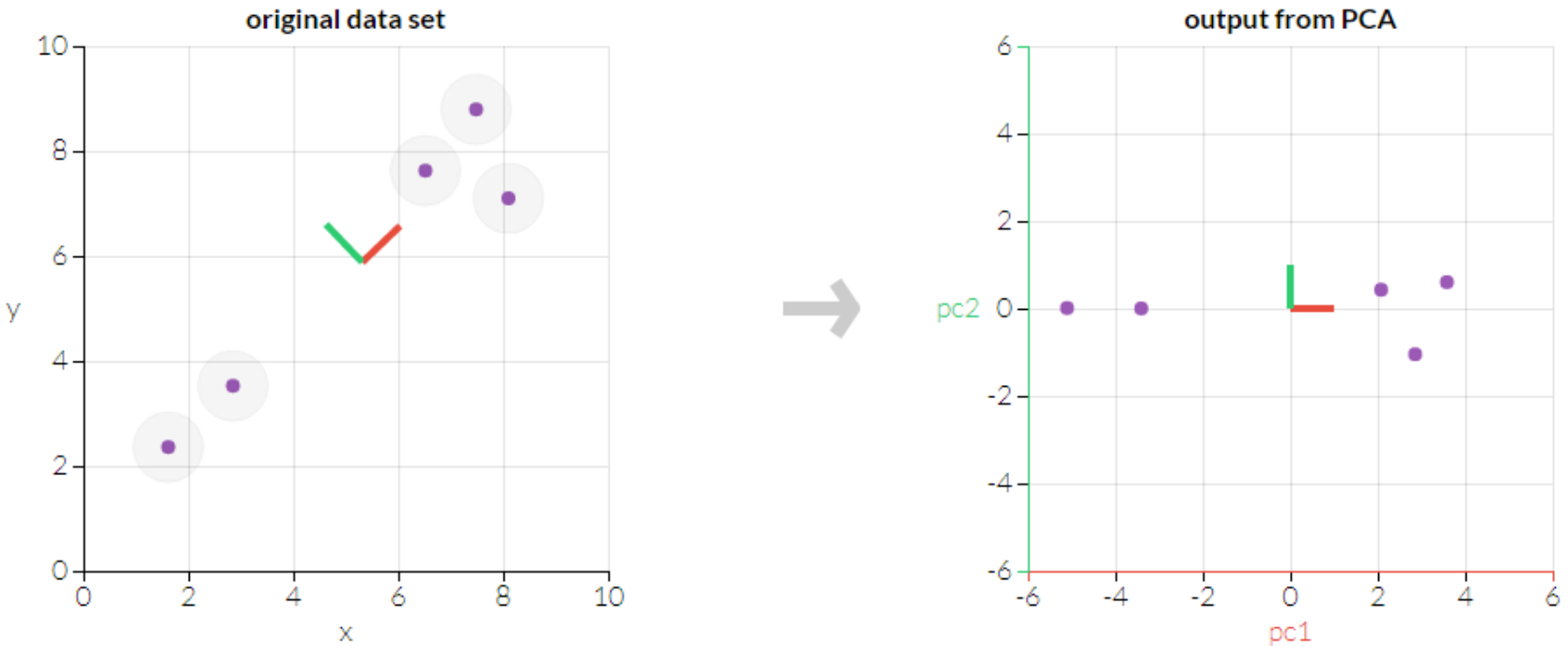
PCA

- Cilj: pronalaženje novog (manjeg) skupa atributa koji bolje predstavlja promenljivost skupa.
- Svaki par novih atributa ima kovarijansu 0.
- Novi atributi su uređeni prema količini promenljivosti koju obuhvataju.
- Prvi (novi) atribut obuhvata najveću promenljivost skupa.
- Svaki sledeći atribut obuhvata promenljivost skupa koja nije pokrivena prethodnim atributima.

PCA

- Novi atributi se zovu glavne komponente .
- Najčešće se je većina promenljivosti skupa pokrivena malim brojem atributa u odnosu na ukupan broj atributa.
- Primena:
  - na skupu manje dimenzije mogu da se primene tehnike koje ne rade dobro sa skupovima velikih dimenzija.
  - vizuelizacija podataka

PCA


original data set
y
x
0
2
4
6
8
10
output from PCA
pc2
pc1
-6
-4
-2
0
2
4
6

PCA u biblioteci scikitlearn

- *sklearn.decomposition.PCA*
- parametri
  - *n_components* - broj komponenti koje će biti sačuvane
- atributi
  - *components_* - Glavne komponente koje predstavljaju pravce maksimalne varijanse u podacima.
  - *explained_variance_* -Količina varijanse koju objašnjava svaka od glavnih komponenti.
  - *explained_variance_ratio_* - Procenat varijanse objašnjen svakom od glavnih komponenti.
- metode
  - *fit*
  - *fit_transform*
  - *transform(X)*
  - *get_covariance()* -računa kovarijanse
  - *inverse_transform(X)*

Outline

1. Veštačke neuronske mreže
2. PCA - Analiza glavnih komponenti (Principal Component Analysis)
3. Zadatak za samostalan rad

Zadatak za samostalan rad

Preuzeti skup podataka *car.csv* o klasama automobila. Koristeći programski jezik Python i neuronske mreže (MLP) izvršiti klasifikaciju nad datim skupom. Atribut *class* sadrži informaciju kojoj klasi pripada automobil. Napraviti različite modele klasifikacije sa različitim arhitekturama neuronske mreže promenom broja skrivenih slojeva i skrivenih čvorova. Broj skrivenih slojeva je u intervalu [1, 3], a broj skrivenih čvorova po sloju je u intervalu [5, 9].

Zadatak za samostalan rad

Izdvojiti

- preciznost na trening skupu;
- preciznost na test skupu;
- matricu konfuzije za trening skup;
- matricu konfuzije za test skup;

Primeniti PCA na skup i izvršiti klasifikaciju nad transformisanim skupom. Koji broj atributa ste izabrali i zašto? Diskutovati dobijene modele.